# Procesos ETL con cursores en SQL Server y paquetes SSIS: de Excel a un modelo relacional y multidimensional

Creadores de oportunidades:

Cristian Mauricio Muñoz

Jeison Crishtofer Rojas

Julieth Alvarado Meriño

BIG DATA 701N SIS BD-A

GESTOR DEL CONOCIMIENTO Y APRENDIZAJE:

Edison Gustavo Cañon

Facultad de Ingeniería

Ingeniería de Sistemas y Computación

Universidad de Cundinamarca

Extensión Chía

Septiembre 2026

**Tabla de contenido**

Introducción . . . . . 4
Objetivos . . . . . 5
Objetivo general . . . . . 5
Objetivos específicos . . . . . 5
Marco conceptual . . . . . 6
Entorno de trabajo . . . . . 7
Diagnóstico inicial de calidad . . . . . 8
Ejercicio de ETL mediante cursores en SQL Server . . . . . 11
Diseño del proceso . . . . . 11
Resultados del cursor . . . . . 12
Código completo del cursor . . . . . 15
Modelo relacional . . . . . 20
Ajustes al modelo para conservar la normalización . . . . . 22
Catálogos de categorías . . . . . 23
Correspondencia entre el Excel y el modelo relacional . . . . . 24
Reglas de validación y tratamiento . . . . . 26
Destinos de los registros . . . . . 26
Reglas aplicadas . . . . . 27
Resolución de duplicados y justificación . . . . . 28
Paquete SSIS A: Excel hacia el modelo relacional . . . . . 28
Carga del Excel al staging . . . . . 29
Del staging al modelo relacional . . . . . 32
Configuración de los componentes de transformación . . . . . 37

Paquete SSIS B: modelo relacional hacia la bodega de datos . . . . . 51

Resultados de la carga . . . . . 55

Motivos de rechazo y de revisión . . . . . 56

Correcciones registradas . . . . . 57

Evidencias antes y después del tratamiento . . . . . 58

Consultas analíticas y verificación . . . . . 60

Segunda ejecución: prueba de no duplicados . . . . . 64

Conclusiones . . . . . 68

Repositorio y video de demostración . . . . . 69

Referencias . . . . . 70

## Introducción

Las organizaciones toman decisiones a partir de datos que casi nunca llegan limpios: los registros vienen con valores faltantes, formatos distintos para un mismo dato, categorías escritas de varias maneras, fechas imposibles y registros repetidos. Rahm y Do (2000) agrupan estos problemas en errores de esquema y de instancia, y muestran que la limpieza de datos es una etapa obligatoria antes de integrar información en una bodega de datos. Esa limpieza se realiza dentro de los procesos de extracción, transformación y carga (ETL), que llevan los datos desde sus fuentes hasta modelos preparados para el análisis (Kimball y Ross, 2013).

Este informe documenta la construcción de esos procesos sobre el archivo *PersonasETLActividad4.xlsx*, un conjunto de 1.000 registros ficticios de personas encuestadas con 37 atributos de identificación, ubicación, educación, empleo, ingresos y hogar. El trabajo se desarrolló en tres partes. En la primera, se implementó en SQL Server un proceso de limpieza y cálculo del balance mensual mediante un cursor de Transact-SQL para los registros 1 a 100. En la segunda, se construyó un paquete de SQL Server Integration Services (SSIS) que valida, corrige, deduplica y carga los 1.000 registros en un modelo relacional normalizado. En la tercera, un segundo paquete SSIS alimenta, desde el modelo depurado, una bodega de datos en esquema estrella sobre la que se resolvieron cuatro consultas analíticas.

El principio que guió todas las decisiones fue no inventar información: un documento, una fecha o un importe desconocido nunca se reemplazó por un valor supuesto. Por eso cada registro termina en uno de cuatro destinos (rechazado, duplicado descartado, en revisión o válido) y todo cambio queda registrado con su motivo. El Excel original se conservó intacto y las transformaciones se hicieron exclusivamente con componentes nativos de SSIS.

El documento presenta primero los objetivos, los conceptos de base y el entorno de trabajo. Luego describe el diagnóstico de calidad, el ejercicio con cursores, el modelo relacional, las reglas de tratamiento y los dos paquetes SSIS, con las evidencias de su ejecución. Cierra con las consultas analíticas, la prueba de re-ejecución sin duplicados y las

conclusiones.

Esta versión incorpora las observaciones de la revisión del docente: el código completo del cursor dentro del informe, las categorías (tipo de vivienda, estado civil, zona y las demás) como tablas de catálogo del modelo relacional, la carga en dos pasos, primero del Excel al staging y luego del staging al modelo relacional, la verificación en SQL de la tabla donde queda el Excel y las capturas de la configuración de los componentes que transforman los datos.

## Objetivos

### Objetivo general

Implementar procesos de extracción, transformación y carga mediante cursores en SQL Server y componentes de SSIS, aplicando tratamientos de calidad para trasladar información desde Excel hacia un modelo relacional y, posteriormente, hacia un modelo multidimensional.

### Objetivos específicos

- Limpiar y validar con un cursor los registros 1 a 100 y calcular su balance mensual sin reemplazar importes desconocidos.
- Diagnosticar la calidad del archivo de origen: valores faltantes, formatos inconsistentes, datos inválidos y duplicados.
- Diseñar un modelo relacional normalizado y cargarlo desde Excel con un paquete SSIS que aplique reglas de validación y registre cambios y rechazos.
- Construir una bodega de datos en esquema estrella con claves sustitutas y cargarla desde el modelo depurado con un segundo paquete SSIS.
- Verificar que las consultas del modelo multidimensional coinciden con el relacional y que la re-ejecución de los paquetes no genera duplicados.

## Marco conceptual

**Proceso ETL.** Es el conjunto de tareas que extrae datos de las fuentes, los transforma para cumplir reglas de calidad y de negocio, y los carga en un destino analítico. Kimball y Ross (2013) recomiendan cargar primero las dimensiones y después los hechos, y resolver la relación entre ambos mediante claves sustitutas generadas por la bodega.

**Calidad de datos.** Rahm y Do (2000) clasifican los problemas de una sola fuente en errores de esquema (por ejemplo, ausencia de restricciones de integridad) y de instancia (valores faltantes, ilegales, mal escritos, contradictorios o duplicados). Todos estos tipos aparecieron en el archivo analizado.

**Normalización.** Un modelo en tercera forma normal elimina las dependencias transitivas: cada atributo no clave depende solo de la clave de su entidad (Elmasri y Navathe, 2016). Por eso los nombres de departamento, municipio y ocupación, y todas las categorías cerradas, se trasladaron a tablas de catálogo.

**Cursores en Transact-SQL.** Un cursor permite recorrer el resultado de una consulta fila por fila con las instrucciones DECLARE, OPEN, FETCH, CLOSE y DEALLOCATE (Microsoft, s.f.-a). Resulta útil cuando cada fila requiere una secuencia de validaciones con decisiones propias.

**SQL Server Integration Services.** SSIS construye flujos de datos con transformaciones nativas como Derived Column, Data Conversion, Conditional Split, Lookup, Aggregate, Sort y Row Count (Microsoft, s.f.-b). La transformación Lookup compara cada fila con una tabla de referencia y permite separar las filas con y sin coincidencia (Microsoft, s.f.-c), lo que se aprovechó para validar catálogos y para evitar duplicados.

**Modelo dimensional.** Un esquema estrella organiza una tabla de hechos con medidas numéricas rodeada de dimensiones descriptivas. La granularidad declara qué representa exactamente una fila de hechos y debe definirse antes de diseñar las dimensiones (Kimball y Ross, 2013).

## Entorno de trabajo

La Tabla 1 resume las herramientas utilizadas. Las bases de datos se crearon en una instancia local de SQL Server 2025: *PersonasETL* contiene el staging, el ejercicio con cursores, el modelo relacional y las bitácoras, y *PersonasDW* contiene la bodega de datos.

**Tabla 1**

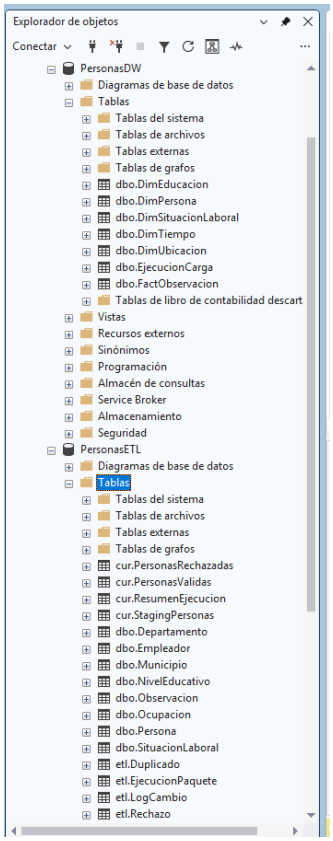
*Herramientas y versiones utilizadas*

| Componente | Versión o detalle |
|---|---|
| Motor de base de datos | SQL Server 2025 (bases PersonasETL y PersonasDW) |
| Administración | SQL Server Management Studio 22 |
| Desarrollo ETL | Visual Studio Community 2026 con la extensión SQL Server Integration Services Projects 2022+, en 64 bits (Microsoft, s.f.-d) |
| Lectura de Excel | Microsoft Access Database Engine 2016 x64 (proveedor ACE OLE DB) |
| Conexión a SQL Server | Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL) |

La Figura 1 muestra las tablas creadas en ambas bases de datos con los scripts del repositorio.

**Figura 1**

*Tablas creadas en las bases PersonasETL y PersonasDW*




*Nota.* Explorador de objetos de SQL Server Management Studio.

## Diagnóstico inicial de calidad

El diagnóstico se ejecutó sobre la copia fiel del Excel cargada en *cur.StagingPersonas*, con el script de solo lectura *07_diagnostico_inicial.sql*. Todas las celdas del archivo están almacenadas como texto, de modo que cada columna se trató como cadena hasta su conversión explícita. La Tabla 2 resume los hallazgos y la Figura 2 muestra los resultados en SQL Server. Según las consideraciones del dataset, el formato de documento SIM más siete dígitos, los teléfonos que empiezan por 000 y los códigos geográficos internos no se consideraron errores por sí mismos.

**Tabla 2**

*Hallazgos del diagnóstico inicial de calidad*

| Categoría | Hallazgo | Registros |
|---|---|---|
| Faltantes | DocumentNumber vacío (45) o con el texto "SIN-DATO" (23) | 68 |
| Faltantes | FirstName vacío | 22 |
| Faltantes | MonthlyExpenses vacío | 22 |
| Faltantes | HealthRegime con "NULL", "sin dato" o "N/A" | 22 |
| Faltantes (opcionales) | MiddleName / SecondLastName vacíos | 301 / 77 |
| Formato | DocumentType en minúscula (cc, ce, pa) | 23 |
| Formato | FirstName en minúscula / LastName en mayúscula | 23 / 23 |
| Formato | MaritalStatus y EducationLevel en minúscula | 44 / 44 |
| Formato | MunicipalityName en minúscula y sin tilde | 22 |
| Formato | BirthDate en formato dd/mm/aaaa | 68 |
| Formato | Phone con guion (000-0000004) | 45 |
| Formato | MonthlyIncome como "$ 1.200.000,00" | 22 |
| Formato | MonthlyExpenses con coma decimal (11200000,50) | 22 |
| Formato | Disability como 0, 1, sí o no | 44 |
| Inválido | BirthDate inexistente (31/02/1990) | 45 |
| Inválido | BirthDate futura (2035-01-01) | 23 |
| Inválido | SurveyDate inexistente (2026-13-40) | 22 |
| Inválido | UpdatedAt anterior a SurveyDate (2020-01-01) | 22 |
| Inválido | EmploymentStartDate futura (2028-01-01) | 22 |
| Inválido | Email sin "@" / teléfono de 3 dígitos | 22 / 22 |
| Inválido | Sex = "Z" / Zone = "CENTRO" / estrato 0, 9 o "ALTO" | 22 / 22 / 22 |
| Inválido | MonthlyIncome en letras ("dos millones") / negativo | 44 / 22 |
| Inválido | Dependents = 2.5, "muchos" o -2 | 44 |
| Inválido | HouseholdSize = 0 | 22 |
| Coherencia | Edad reportada que difiere 12 años o más de la calculada | 46 |
| Coherencia | Municipio desconocido (M999) | 45 |
| Coherencia | Departamento que no corresponde al municipio | 22 |
| Coherencia | EMPLEADO con contrato NO_APLICA | 71 |
| Duplicados | Pares con el mismo documento y la misma SurveyDate | 50 pares |
| Duplicados | Copias idénticas (801-825 frente a 951-975) | 25 |
| Duplicados | Versiones distintas: ingreso +500.000 y UpdatedAt de 12:30 a 18:45 (826-850 frente a 976-1000) | 25 |

**Figura 2**

*Resultados del script de diagnóstico inicial*

Resultados | Mensajes

| | Columna | Faltantes |
|---|---|---|
| 1 | EmployerName | 802 |
| 2 | OccupationCode | 599 |
| 3 | EmploymentStartDate | 596 |
| 4 | MiddleName | 301 |
| 5 | SecondLastName | 77 |
| 6 | DocumentNumber (vacío o SIN-DATO) | 68 |
| 7 | FirstName | 22 |
| 8 | HealthRegime (NULL/sin dato/N/A) | 22 |
| 9 | MonthlyExpenses | 22 |
| 10 | LastName | 0 |

| | Hallazgo | Registros |
|---|---|---|
| 1 | DocumentType en minúscula | 23 |
| 2 | FirstName en minúscula | 23 |
| 3 | LastName en MAYÚSCULA | 23 |
| 4 | MaritalStatus en minúscula | 44 |
| 5 | EducationLevel en minúscula | 44 |
| 6 | MunicipalityName en minús... | 22 |
| 7 | BirthDate con formato dd/m... | 68 |
| 8 | Phone con guion | 45 |
| 9 | MonthlyIncome con $ y sep... | 22 |
| 10 | MonthlyExpenses con com... | 22 |
| 11 | Disability como 0/1/sí/no | 44 |

| | Hallazgo | Registros |
|---|---|---|
| 1 | Documento sin formato SIM+7 dígitos | 68 |
| 2 | BirthDate inexistente (ej. 31/02) | 45 |
| 3 | BirthDate futura | 23 |
| 4 | SurveyDate inválida | 22 |
| 5 | UpdatedAt anterior a SurveyDate | 22 |
| 6 | EmploymentStartDate posterior a Su... | 22 |
| 7 | Email sin @ | 22 |
| 8 | Teléfono de longitud inválida | 22 |
| 9 | Sex fuera de M/F/ND | 22 |
| 10 | Zone fuera de URBANA/RURAL | 22 |
| 11 | Estrato fuera de 1-6 | 22 |
| 12 | MonthlyIncome no numérico | 44 |
| 13 | MonthlyIncome negativo | 22 |
| 14 | Dependents no entero o negativo | 44 |
| 15 | HouseholdSize = 0 | 22 |
| 16 | Municipio desconocido (M999) | 45 |
| 17 | EMPLEADO con contrato NO_APLI... | 71 |
| 18 | EMPLEADO sin ocupación | 44 |

| | Hallazgo | Grupos |
|---|---|---|
| 1 | Personas con más de un registro | 50 |
| 2 | Observaciones duplicadas (persona + SurveyDate) | 50 |

| | FilaA | FilaB | DocumentNumber | SurveyDate | IngresoA | IngresoB | ActualizadoA | ActualizadoB | Tipo |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 801 | 951 | SIM0000801 | 2025-03-27 | 2100000 | 2100000 | 2025-03-27 12:30:00 | 2025-03-27 12:30:00 | Idéntico |
| 2 | 802 | 952 | SIM0000802 | 2026-06-11 | 14200000 | 14200000 | 2026-06-11 12:30:00 | 2026-06-11 12:30:00 | Idéntico |
| 3 | 803 | 953 | SIM0000803 | 2026-08-12 | 6300000 | 6300000 | 2026-08-12 12:30:00 | 2026-08-12 12:30:00 | Idéntico |
| 4 | 804 | 954 | SIM0000804 | 2025-03-27 | 3900000 | 3900000 | 2025-03-27 12:30:00 | 2025-03-27 12:30:00 | Idéntico |
| 5 | 805 | 955 | SIM0000805 | 2026-03-14 | 14800000 | 14800000 | 2026-03-14 12:30:00 | 2026-03-14 12:30:00 | Idéntico |
| 6 | 806 | 956 | SIM0000806 | 2025-06-17 | 10800000 | 10800000 | 2025-06-17 12:30:00 | 2025-06-17 12:30:00 | Idéntico |
| 7 | 807 | 957 | SIM0000807 | 2026-01-10 | 9600000 | 9600000 | 2026-01-10 12:30:00 | 2026-01-10 12:30:00 | Idéntico |
| 8 | 808 | 958 | SIM0000808 | 2026-06-13 | 14000000 | 14000000 | 2026-06-13 12:30:00 | 2026-06-13 12:30:00 | Idéntico |
| 9 | 809 | 959 | SIM0000809 | 2025-02-01 | 4800000 | 4800000 | 2025-02-01 12:30:00 | 2025-02-01 12:30:00 | Idéntico |
| 10 | 810 | 960 | SIM0000810 | 2026-02-20 | 9000000 | 9000000 | 2026-02-20 12:30:00 | 2026-02-20 12:30:00 | Idéntico |
| 11 | 811 | 961 | SIM0000811 | 2025-04-14 | 12300000 | 12300000 | 2025-04-14 12:30:00 | 2025-04-14 12:30:00 | Idéntico |
| 12 | 812 | 962 | SIM0000812 | 2026-03-22 | 12900000 | 12900000 | 2026-03-22 12:30:00 | 2026-03-22 12:30:00 | Idéntico |
| 13 | 813 | 963 | SIM0000813 | 2025-01-10 | 9800000 | 9800000 | 2025-01-10 12:30:00 | 2025-01-10 12:30:00 | Idéntico |
| 14 | 814 | 964 | SIM0000814 | 2026-08-04 | 4900000 | 4900000 | 2026-08-04 12:30:00 | 2026-08-04 12:30:00 | Idéntico |
| 15 | 815 | 965 | SIM0000815 | 2025-07-01 | 10800000 | 10800000 | 2025-07-01 12:30:00 | 2025-07-01 12:30:00 | Idéntico |
| 16 | 816 | 966 | SIM0000816 | 2026-08-01 | 13600000 | 13600000 | 2026-08-01 12:30:00 | 2026-08-01 12:30:00 | Idéntico |
| 17 | 817 | 967 | SIM0000817 | 2026-05-04 | 2000000 | 2000000 | 2026-05-04 12:30:00 | 2026-05-04 12:30:00 | Idéntico |
| 18 | 818 | 968 | SIM0000818 | 2026-08-15 | 6800000 | 6800000 | 2026-08-15 12:30:00 | 2026-08-15 12:30:00 | Idéntico |
| 19 | 819 | 969 | SIM0000819 | 2025-07-04 | 12300000 | 12300000 | 2025-07-04 12:30:00 | 2025-07-04 12:30:00 | Idéntico |
| 20 | 820 | 970 | SIM0000820 | 2025-04-05 | 10500000 | 10500000 | 2025-04-05 12:30:00 | 2025-04-05 12:30:00 | Idéntico |
| 21 | 821 | 971 | SIM0000821 | 2025-04-04 | 6000000 | 6000000 | 2025-04-04 12:30:00 | 2025-04-04 12:30:00 | Idéntico |
| 22 | 822 | 972 | SIM0000822 | 2025-02-05 | 10400000 | 10400000 | 2025-02-05 12:30:00 | 2025-02-05 12:30:00 | Idéntico |

Consulta ejecutada correctamente.

*Nota.* De arriba abajo: valores faltantes y formatos inconsistentes, datos inválidos y duplicados.

## Ejercicio de ETL mediante cursores en SQL Server

### Diseño del proceso

El script *02_cursor_cargar_staging.sql* copió las 1.000 filas en *cur.StagingPersonas* conservando los valores originales: treinta y siete columnas de texto, con celda vacía como NULL y texto vacío como cadena vacía. El cursor, en *03_cursor_etl.sql*, recorre solo los registros con SourceRowId entre 1 y 100 y aplica a cada uno los siguientes pasos:

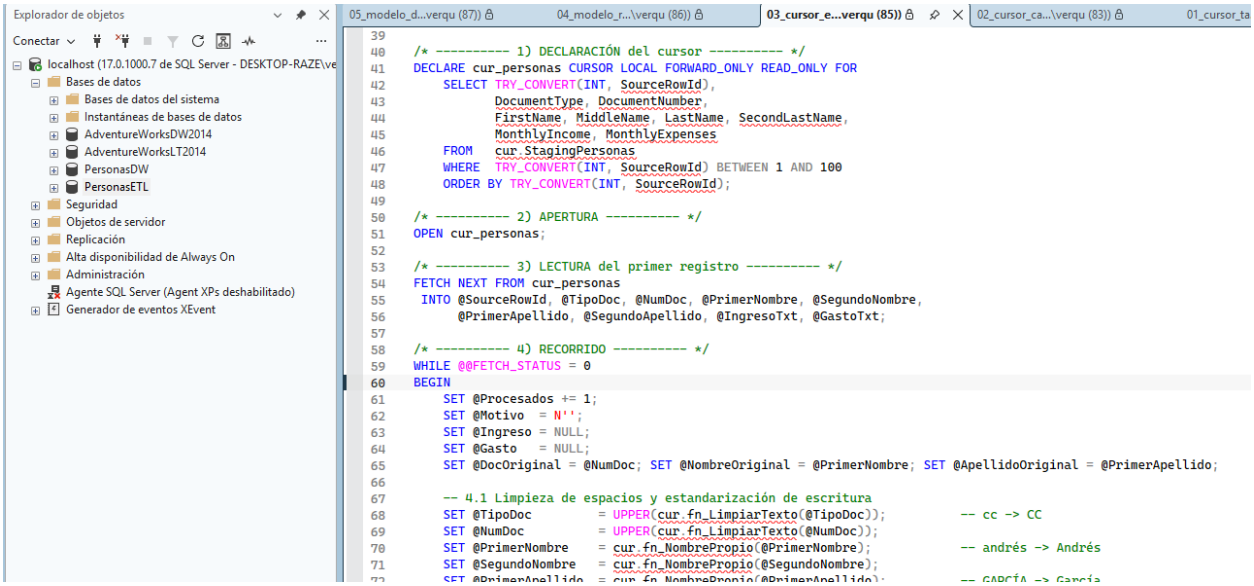
- Limpieza de espacios con *cur.fn_LimpiarTexto*, tipo de documento en mayúsculas y nombres y apellidos en formato de nombre propio con *cur.fn_NombrePropio*.
- Identificación de registros sin documento (vacío o "SIN-DATO"), sin primer nombre o sin primer apellido.
- Conversión de MonthlyIncome y MonthlyExpenses: se eliminan "$", espacios y punto de miles, la coma decimal pasa a punto y se aplica TRY_CONVERT a DECIMAL(18,2).
- Detección de importes faltantes, no numéricos o negativos, que nunca se reemplazan por cero.

- Cálculo del balance mensual como ingreso menos gasto. Un balance negativo es válido.
- Inserción en *cur.PersonasValidas* o en *cur.PersonasRechazadas* con todos los motivos del rechazo.

El código usa las seis instrucciones pedidas: DECLARE (declaración), OPEN (apertura), FETCH NEXT (lectura), WHILE @@FETCH_STATUS = 0 (recorrido), CLOSE (cierre) y DEALLOCATE (liberación). La Figura 3 muestra el fragmento con la declaración, la apertura y la lectura, y al final de esta sección se presenta el código completo.

**Figura 3**

*Declaración, apertura y lectura del cursor en SQL Server Management Studio*




**Resultados del cursor**

El cursor procesó 100 registros: aceptó 78 y rechazó 22 (Tabla 3). Entre los aceptados, 30 tienen balance negativo, lo que confirma que esa condición no causa rechazo. Un registro puede acumular más de un motivo; por ejemplo, la fila 31 no tiene documento y la fila 13 tiene el ingreso escrito en letras.

**Tabla 3**

*Resultados del proceso con cursor (SourceRowId 1 a 100)*

| Concepto | Registros |
|---|---|
| Procesados | 100 |
| Aceptados | 78 |
| Rechazados | 22 |
| Aceptados con balance negativo | 30 |
| Motivo: Sin documento | 8 |
| Motivo: Ingreso no numérico | 6 |
| Motivo: Ingreso negativo | 3 |
| Motivo: Gasto faltante | 3 |
| Motivo: Sin primer nombre | 2 |

*Nota.* La suma de motivos supera el número de rechazados porque algunas filas tienen varios motivos.

La Tabla 4 compara los importes originales con los convertidos en una muestra de registros válidos. La fila 18 convierte "$ 14.700.000,00" en 14.700.000,00, y la fila 22 convierte "8900000,50", con coma decimal, en 8.900.000,50.

**Tabla 4**

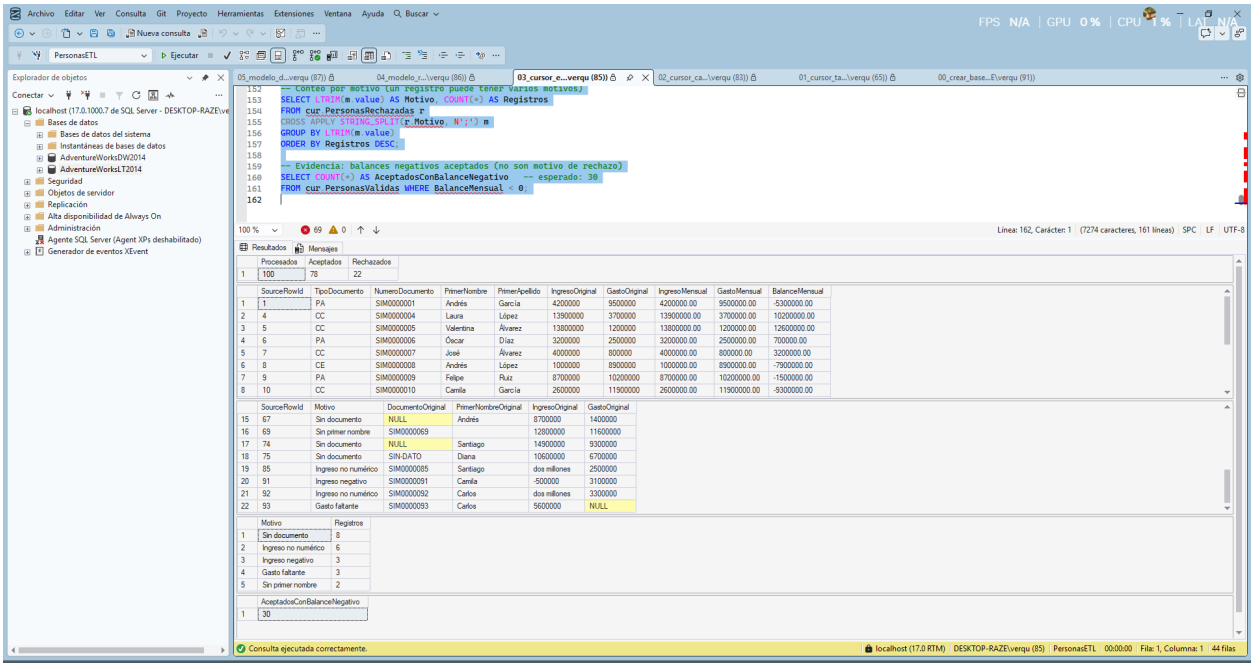
*Muestra de registros válidos antes y después de la conversión*

| Fila | Documento | Nombre | Ingreso original | Gasto original | Ingreso | Gasto | Balance |
|---|---|---|---|---|---|---|---|
| 1 | PA SIM0000001 | Andrés García | 4200000 | 9500000 | 4.200.000,00 | 9.500.000,00 | -5.300.000,00 |
| 4 | CC SIM0000004 | Laura López | 13900000 | 3700000 | 13.900.000,00 | 3.700.000,00 | 10.200.000,00 |
| 8 | CE SIM0000008 | Andrés López | 1000000 | 8900000 | 1.000.000,00 | 8.900.000,00 | -7.900.000,00 |
| 18 | CC SIM0000018 | Valentina Gómez | $ 14.700.000,00 | 2600000 | 14.700.000,00 | 2.600.000,00 | 12.100.000,00 |
| 22 | CC SIM0000022 | Valentina López | 5300000 | 8900000,50 | 5.300.000,00 | 8.900.000,50 | -3.600.000,50 |
| 30 | CC SIM0000030 | Santiago Gómez | 7500000 | 5900000 | 7.500.000,00 | 5.900.000,00 | 1.600.000,00 |

La Figura 4 muestra la ejecución completa del cursor en SQL Server, y la Figura 5 detalla los registros válidos con su importe original, el convertido y el balance.

**Figura 4**

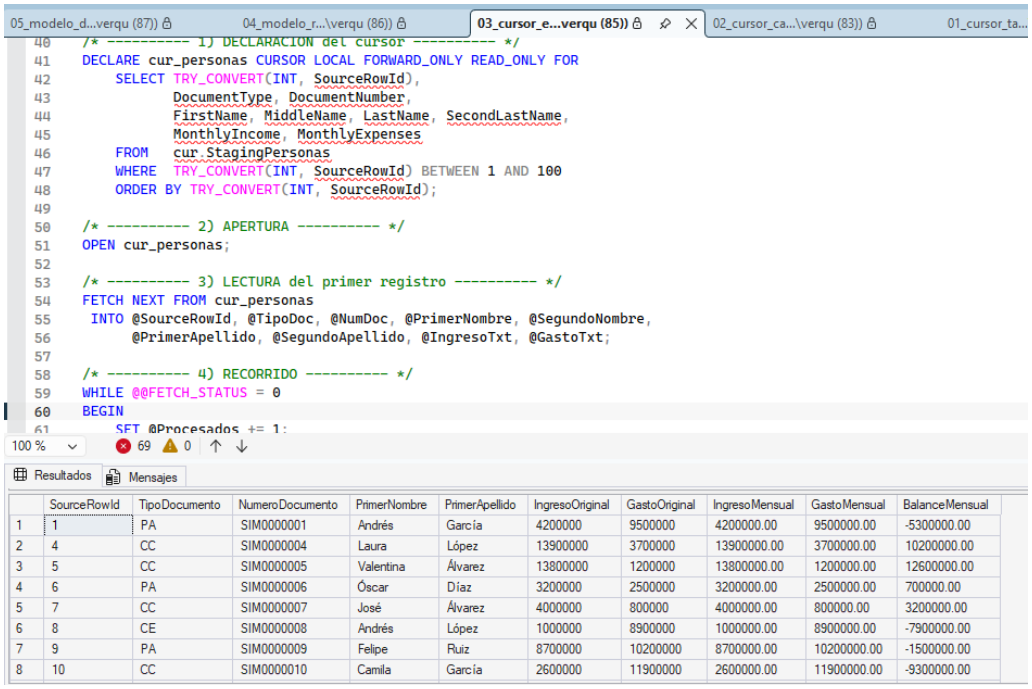
*Ejecución del cursor: conteos, registros válidos, registros rechazados y motivos*




*Nota.* Los avisos rojos de la barra inferior del editor son subrayados del corrector IntelliSense, no errores de ejecución; la consulta terminó correctamente.

**Figura 5**

*Registros válidos con el importe original, el convertido y el balance mensual*



```
/* ---------- 1) DECLARACIÓN del cursor ---------- */
DECLARE cur_personas CURSOR LOCAL FORWARD_ONLY READ_ONLY FOR
    SELECT TRY_CONVERT(INT, SourceRowId),
           DocumentType, DocumentNumber,
           FirstName, MiddleName, LastName, SecondLastName,
           MonthlyIncome, MonthlyExpenses
    FROM   cur.StagingPersonas
    WHERE  TRY_CONVERT(INT, SourceRowId) BETWEEN 1 AND 100
    ORDER BY TRY_CONVERT(INT, SourceRowId);

/* ---------- 2) APERTURA ---------- */
OPEN cur_personas;

/* ---------- 3) LECTURA del primer registro ---------- */
FETCH NEXT FROM cur_personas
 INTO @SourceRowId, @TipoDoc, @NumDoc, @PrimerNombre, @SegundoNombre,
      @PrimerApellido, @SegundoApellido, @IngresoTxt, @GastoTxt;

/* ---------- 4) RECORRIDO ---------- */
WHILE @@FETCH_STATUS = 0
BEGIN
    SET @Procesados += 1;
```

| | SourceRowId | TipoDocumento | NumeroDocumento | PrimerNombre | PrimerApellido | IngresoOriginal | GastoOriginal | IngresoMensual | GastoMensual | BalanceMensual |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | PA | SIM0000001 | Andrés | García | 4200000 | 9500000 | 4200000.00 | 9500000.00 | -5300000.00 |
| 2 | 4 | CC | SIM0000004 | Laura | López | 13900000 | 3700000 | 13900000.00 | 3700000.00 | 10200000.00 |
| 3 | 5 | CC | SIM0000005 | Valentina | Álvarez | 13800000 | 1200000 | 13800000.00 | 1200000.00 | 12600000.00 |
| 4 | 6 | PA | SIM0000006 | Óscar | Díaz | 3200000 | 2500000 | 3200000.00 | 2500000.00 | 700000.00 |
| 5 | 7 | CC | SIM0000007 | José | Álvarez | 4000000 | 800000 | 4000000.00 | 800000.00 | 3200000.00 |
| 6 | 8 | CE | SIM0000008 | Andrés | López | 1000000 | 8900000 | 1000000.00 | 8900000.00 | -7900000.00 |
| 7 | 9 | PA | SIM0000009 | Felipe | Ruiz | 8700000 | 10200000 | 8700000.00 | 10200000.00 | -1500000.00 |
| 8 | 10 | CC | SIM0000010 | Camila | García | 2600000 | 11900000 | 2600000.00 | 11900000.00 | -9300000.00 |

## Código completo del cursor

A continuación se transcriben completos los dos scripts del ejercicio. El primero, *01_cursor_tablas.sql*, crea la tabla de staging, las funciones de limpieza y las tablas de destino; el segundo, *03_cursor_etl.sql*, contiene el cursor con sus seis instrucciones y las consultas de resultados. El script *02_cursor_cargar_staging.sql* no se transcribe porque solo contiene los 1.000 INSERT con los valores del Excel.

## Archivo 01_cursor_tablas.sql

```
/* ====================================================================
   01_cursor_tablas.sql
   Parte 1 - Ejercicio de ETL con cursores.
   Crea: tabla de staging (valores originales, todo como texto),
         funciones de apoyo para limpieza de texto,
         tablas de destino (válidos / rechazados) y resumen de ejecución.
   ==================================================================== */
USE PersonasETL;
GO

/* ---------- 1. Staging: copia fiel del Excel (todo NVARCHAR, sin limpiar) ---------- */
DROP TABLE IF EXISTS cur.StagingPersonas;
CREATE TABLE cur.StagingPersonas (
    SourceRowId          NVARCHAR(20)  NULL,
    DocumentType         NVARCHAR(50)  NULL,
    DocumentNumber       NVARCHAR(50)  NULL,
    FirstName            NVARCHAR(100) NULL,
```

```
    MiddleName           NVARCHAR(100) NULL,
    LastName             NVARCHAR(100) NULL,
    SecondLastName       NVARCHAR(100) NULL,
    BirthDate            NVARCHAR(50)  NULL,
    ReportedAge          NVARCHAR(50)  NULL,
    Sex                  NVARCHAR(50)  NULL,
    MaritalStatus        NVARCHAR(50)  NULL,
    Email                NVARCHAR(200) NULL,
    Phone                NVARCHAR(50)  NULL,
    DepartmentCode       NVARCHAR(50)  NULL,
    DepartmentName       NVARCHAR(100) NULL,
    MunicipalityCode     NVARCHAR(50)  NULL,
    MunicipalityName     NVARCHAR(100) NULL,
    Zone                 NVARCHAR(50)  NULL,
    Address              NVARCHAR(200) NULL,
    HousingType          NVARCHAR(50)  NULL,
    SocioeconomicStratum NVARCHAR(50)  NULL,
    EducationLevel       NVARCHAR(50)  NULL,
    EmploymentStatus     NVARCHAR(50)  NULL,
    OccupationCode       NVARCHAR(50)  NULL,
    OccupationName       NVARCHAR(100) NULL,
    EmployerName         NVARCHAR(200) NULL,
    ContractType         NVARCHAR(50)  NULL,
    EmploymentStartDate  NVARCHAR(50)  NULL,
    MonthlyIncome        NVARCHAR(50)  NULL,
    MonthlyExpenses      NVARCHAR(50)  NULL,
    Dependents           NVARCHAR(50)  NULL,
    HouseholdSize        NVARCHAR(50)  NULL,
    HealthRegime         NVARCHAR(50)  NULL,
    Disability           NVARCHAR(50)  NULL,
    SurveyDate           NVARCHAR(50)  NULL,
    UpdatedAt            NVARCHAR(50)  NULL,
    SourceChannel        NVARCHAR(50)  NULL,
    FechaCargaStaging    DATETIME2(0)  NOT NULL DEFAULT SYSDATETIME()
);
GO

/* ---------- 2. Funciones de apoyo ---------- */
-- Quita espacios al inicio/fin, colapsa espacios internos repetidos y convierte '' en NULL.
CREATE OR ALTER FUNCTION cur.fn_LimpiarTexto (@texto NVARCHAR(400))
RETURNS NVARCHAR(400)
AS
BEGIN
    DECLARE @t NVARCHAR(400) = REPLACE(REPLACE(@texto, NCHAR(160), N' '), NCHAR(9), N' ');
    SET @t = TRIM(@t);
    WHILE CHARINDEX(N'  ', @t) > 0
        SET @t = REPLACE(@t, N'  ', N' ');
    RETURN NULLIF(@t, N'');
END;
GO

-- Escritura tipo "Nombre Propio": primera letra de cada palabra en mayúscula.
CREATE OR ALTER FUNCTION cur.fn_NombrePropio (@texto NVARCHAR(400))
RETURNS NVARCHAR(400)
AS
BEGIN
    DECLARE @t NVARCHAR(400) = LOWER(cur.fn_LimpiarTexto(@texto));
    IF @t IS NULL RETURN NULL;
    DECLARE @i INT = 1, @n INT = LEN(@t), @resultado NVARCHAR(400) = N'', @anterior NCHAR(1) = N' ';
    WHILE @i <= @n
    BEGIN
        SET @resultado += CASE WHEN @anterior = N' ' THEN UPPER(SUBSTRING(@t, @i, 1))
                               ELSE SUBSTRING(@t, @i, 1) END;
        SET @anterior = SUBSTRING(@t, @i, 1);
        SET @i += 1;
    END;
    RETURN @resultado;
END;
GO

/* ---------- 3. Tablas de destino ---------- */
DROP TABLE IF EXISTS cur.PersonasValidas;
```

```
CREATE TABLE cur.PersonasValidas (
    SourceRowId          INT           NOT NULL PRIMARY KEY,
    TipoDocumento        NVARCHAR(5)   NOT NULL,
    NumeroDocumento      NVARCHAR(20)  NOT NULL,
    PrimerNombre         NVARCHAR(60)  NOT NULL,
    SegundoNombre        NVARCHAR(60)  NULL,
    PrimerApellido       NVARCHAR(60)  NOT NULL,
    SegundoApellido      NVARCHAR(60)  NULL,
    IngresoOriginal      NVARCHAR(50)  NULL,   -- evidencia "antes"
    GastoOriginal        NVARCHAR(50)  NULL,
    IngresoMensual       DECIMAL(18,2) NOT NULL,
    GastoMensual         DECIMAL(18,2) NOT NULL,
    BalanceMensual       DECIMAL(18,2) NOT NULL,   -- puede ser negativo (válido)
    FechaProceso         DATETIME2(0)  NOT NULL DEFAULT SYSDATETIME()
);

DROP TABLE IF EXISTS cur.PersonasRechazadas;
CREATE TABLE cur.PersonasRechazadas (
    SourceRowId          INT           NOT NULL PRIMARY KEY,
    Motivo               NVARCHAR(500) NOT NULL,
    DocumentoOriginal    NVARCHAR(50)  NULL,
    PrimerNombreOriginal NVARCHAR(100) NULL,
    PrimerApellidoOriginal NVARCHAR(100) NULL,
    IngresoOriginal      NVARCHAR(50)  NULL,
    GastoOriginal        NVARCHAR(50)  NULL,
    FechaProceso         DATETIME2(0)  NOT NULL DEFAULT SYSDATETIME()
);

DROP TABLE IF EXISTS cur.ResumenEjecucion;
CREATE TABLE cur.ResumenEjecucion (
    EjecucionId   INT IDENTITY(1,1) PRIMARY KEY,
    FechaEjecucion DATETIME2(0) NOT NULL DEFAULT SYSDATETIME(),
    Procesados    INT NOT NULL,
    Aceptados     INT NOT NULL,
    Rechazados    INT NOT NULL
);
GO
PRINT N'Tablas del ejercicio con cursor creadas.';
```

## Archivo 03_cursor_etl.sql

```
/* ====================================================================
   03_cursor_etl.sql
   Parte 1 - ETL con cursor sobre SourceRowId 1..100.
   Pasos por registro:
     1. Limpieza de espacios y estandarización (tipo doc., nombres, apellidos)
     2. Identificación de faltantes: documento, primer nombre, primer apellido
     3. Conversión de MonthlyIncome / MonthlyExpenses a número
        (quita '$', espacios y '.' de miles; ',' decimal -> '.')
     4. Detección de faltantes, no numéricos y negativos
     5. Balance mensual = ingreso - gasto (un balance negativo ES válido)
     6. Inserción en cur.PersonasValidas o cur.PersonasRechazadas (con motivo)
   Regla: NUNCA se reemplaza un importe desconocido por 0.
   Instrucciones del cursor demostradas: DECLARE, OPEN, FETCH (lectura),
   WHILE @@FETCH_STATUS (recorrido), CLOSE, DEALLOCATE.
   ==================================================================== */
USE PersonasETL;
GO
SET NOCOUNT ON;

-- El proceso es repetible: se limpian los destinos antes de cada corrida
TRUNCATE TABLE cur.PersonasValidas;
TRUNCATE TABLE cur.PersonasRechazadas;

/* ---------- Variables de trabajo ---------- */
DECLARE @SourceRowId     INT,
        @TipoDoc         NVARCHAR(50),
        @NumDoc          NVARCHAR(50),
        @PrimerNombre    NVARCHAR(100),
        @SegundoNombre   NVARCHAR(100),
```

```
        @PrimerApellido  NVARCHAR(100),
        @SegundoApellido NVARCHAR(100),
        @IngresoTxt      NVARCHAR(50),
        @GastoTxt        NVARCHAR(50);

DECLARE @DocOriginal NVARCHAR(50), @NombreOriginal NVARCHAR(100), @ApellidoOriginal NVARCHAR(100),
        @Ingreso DECIMAL(18,2), @Gasto DECIMAL(18,2), @Texto NVARCHAR(50), @Motivo NVARCHAR(500);

DECLARE @Procesados INT = 0, @Aceptados INT = 0, @Rechazados INT = 0;

/* ---------- 1) DECLARACIÓN del cursor ---------- */
DECLARE cur_personas CURSOR LOCAL FORWARD_ONLY READ_ONLY FOR
    SELECT TRY_CONVERT(INT, SourceRowId),
           DocumentType, DocumentNumber,
           FirstName, MiddleName, LastName, SecondLastName,
           MonthlyIncome, MonthlyExpenses
    FROM   cur.StagingPersonas
    WHERE  TRY_CONVERT(INT, SourceRowId) BETWEEN 1 AND 100
    ORDER BY TRY_CONVERT(INT, SourceRowId);

/* ---------- 2) APERTURA ---------- */
OPEN cur_personas;

/* ---------- 3) LECTURA del primer registro ---------- */
FETCH NEXT FROM cur_personas
 INTO @SourceRowId, @TipoDoc, @NumDoc, @PrimerNombre, @SegundoNombre,
      @PrimerApellido, @SegundoApellido, @IngresoTxt, @GastoTxt;

/* ---------- 4) RECORRIDO ---------- */
WHILE @@FETCH_STATUS = 0
BEGIN
    SET @Procesados += 1;
    SET @Motivo  = N'';
    SET @Ingreso = NULL;
    SET @Gasto   = NULL;
    SET @DocOriginal = @NumDoc; SET @NombreOriginal = @PrimerNombre; SET @ApellidoOriginal = @PrimerApellido;

    -- 4.1 Limpieza de espacios y estandarización de escritura
    SET @TipoDoc         = UPPER(cur.fn_LimpiarTexto(@TipoDoc));          -- cc -> CC
    SET @NumDoc          = UPPER(cur.fn_LimpiarTexto(@NumDoc));
    SET @PrimerNombre    = cur.fn_NombrePropio(@PrimerNombre);            -- andrés -> Andrés
    SET @SegundoNombre   = cur.fn_NombrePropio(@SegundoNombre);
    SET @PrimerApellido  = cur.fn_NombrePropio(@PrimerApellido);          -- GARCÍA -> García
    SET @SegundoApellido = cur.fn_NombrePropio(@SegundoApellido);

    -- 4.2 Campos obligatorios ('SIN-DATO' equivale a no tener documento)
    IF @NumDoc IS NULL OR @NumDoc = N'SIN-DATO' SET @Motivo += N'Sin documento; ';
    IF @PrimerNombre   IS NULL                  SET @Motivo += N'Sin primer nombre; ';
    IF @PrimerApellido IS NULL                  SET @Motivo += N'Sin primer apellido; ';

    -- 4.3 Ingreso mensual: limpiar símbolos y separadores, luego convertir
    SET @Texto = cur.fn_LimpiarTexto(@IngresoTxt);
    IF @Texto IS NULL
        SET @Motivo += N'Ingreso faltante; ';
    ELSE
    BEGIN
        SET @Texto   = REPLACE(REPLACE(REPLACE(@Texto, N'$', N''), N' ', N''), N'.', N'');  -- '.' = miles
        SET @Texto   = REPLACE(@Texto, N',', N'.');                                        -- ',' = decimal
        SET @Ingreso = TRY_CONVERT(DECIMAL(18,2), @Texto);
        IF @Ingreso IS NULL     SET @Motivo += N'Ingreso no numérico; ';
        ELSE IF @Ingreso < 0    SET @Motivo += N'Ingreso negativo; ';
    END;

    -- 4.4 Gasto mensual: misma regla
    SET @Texto = cur.fn_LimpiarTexto(@GastoTxt);
    IF @Texto IS NULL
        SET @Motivo += N'Gasto faltante; ';
    ELSE
    BEGIN
        SET @Texto = REPLACE(REPLACE(REPLACE(@Texto, N'$', N''), N' ', N''), N'.', N'');
        SET @Texto = REPLACE(@Texto, N',', N'.');
        SET @Gasto = TRY_CONVERT(DECIMAL(18,2), @Texto);
```

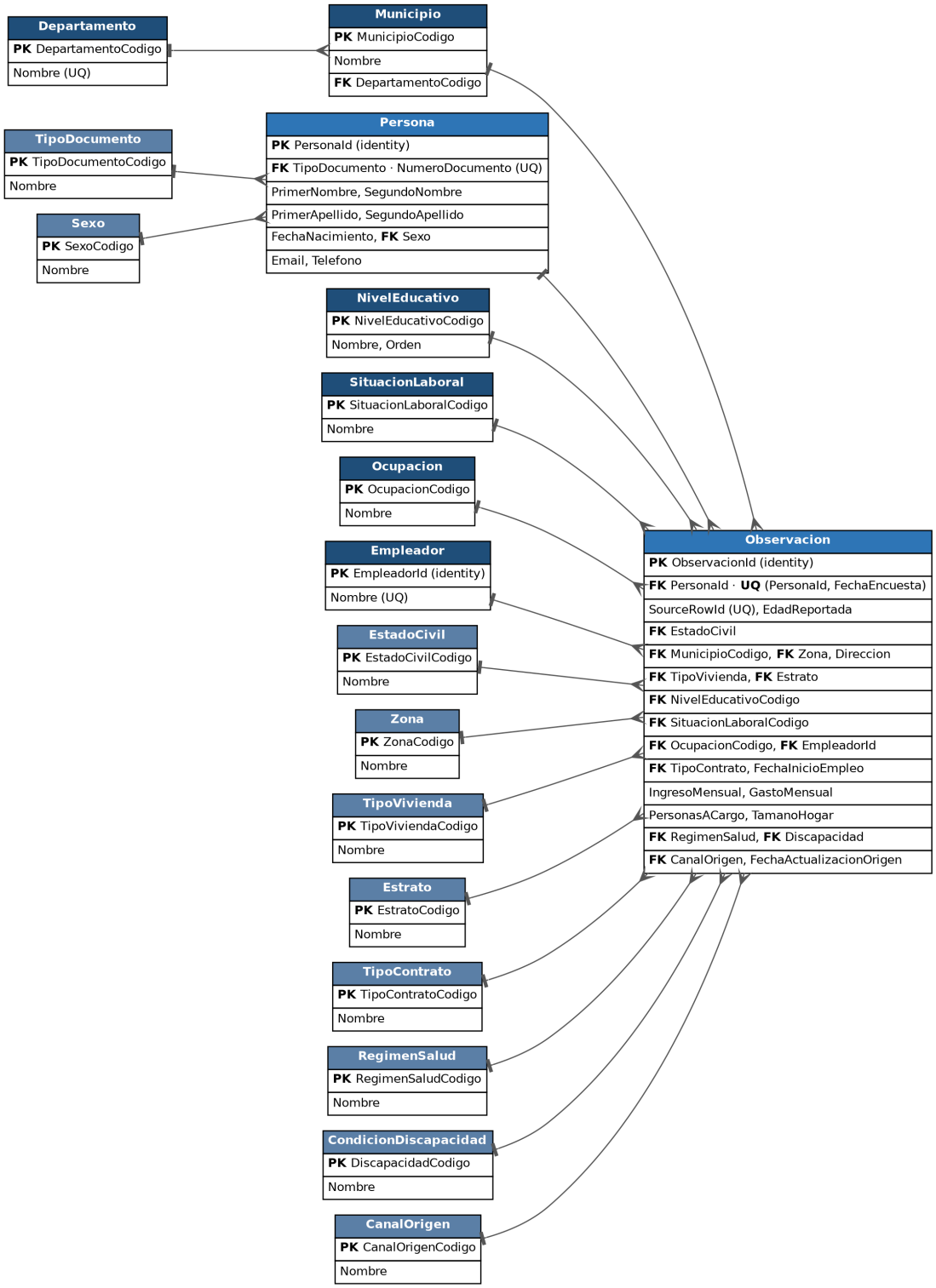
```
        IF @Gasto IS NULL       SET @Motivo += N'Gasto no numérico; ';
        ELSE IF @Gasto < 0      SET @Motivo += N'Gasto negativo; ';
    END;

    -- 4.5 Destino: válido (con balance) o rechazado (con motivo)
    IF @Motivo = N''
    BEGIN
        INSERT INTO cur.PersonasValidas
              (SourceRowId, TipoDocumento, NumeroDocumento, PrimerNombre, SegundoNombre,
               PrimerApellido, SegundoApellido, IngresoOriginal, GastoOriginal,
               IngresoMensual, GastoMensual, BalanceMensual)
        VALUES (@SourceRowId, @TipoDoc, @NumDoc, @PrimerNombre, @SegundoNombre,
                @PrimerApellido, @SegundoApellido, @IngresoTxt, @GastoTxt,
                @Ingreso, @Gasto, @Ingreso - @Gasto);
        SET @Aceptados += 1;
    END
    ELSE
    BEGIN
        INSERT INTO cur.PersonasRechazadas
              (SourceRowId, Motivo, DocumentoOriginal, PrimerNombreOriginal,
               PrimerApellidoOriginal, IngresoOriginal, GastoOriginal)
        VALUES (@SourceRowId, LEFT(@Motivo, LEN(@Motivo) - 1),   -- quita el último ';'
                @DocOriginal, @NombreOriginal, @ApellidoOriginal, @IngresoTxt, @GastoTxt);
        SET @Rechazados += 1;
    END;

    -- Lectura del siguiente registro
    FETCH NEXT FROM cur_personas
     INTO @SourceRowId, @TipoDoc, @NumDoc, @PrimerNombre, @SegundoNombre,
          @PrimerApellido, @SegundoApellido, @IngresoTxt, @GastoTxt;
END;

/* ---------- 5) CIERRE y 6) LIBERACIÓN ---------- */
CLOSE cur_personas;
DEALLOCATE cur_personas;

/* ---------- Resultados ---------- */
INSERT INTO cur.ResumenEjecucion (Procesados, Aceptados, Rechazados)
VALUES (@Procesados, @Aceptados, @Rechazados);

SELECT @Procesados AS Procesados, @Aceptados AS Aceptados, @Rechazados AS Rechazados;
-- Esperado: Procesados 100 | Aceptados 78 | Rechazados 22

SELECT TOP (15) SourceRowId, TipoDocumento, NumeroDocumento, PrimerNombre, PrimerApellido,
       IngresoOriginal, GastoOriginal, IngresoMensual, GastoMensual, BalanceMensual
FROM cur.PersonasValidas ORDER BY SourceRowId;

SELECT SourceRowId, Motivo, DocumentoOriginal, PrimerNombreOriginal, IngresoOriginal, GastoOriginal
FROM cur.PersonasRechazadas ORDER BY SourceRowId;

-- Conteo por motivo (un registro puede tener varios motivos)
SELECT LTRIM(m.value) AS Motivo, COUNT(*) AS Registros
FROM cur.PersonasRechazadas r
CROSS APPLY STRING_SPLIT(r.Motivo, N';') m
GROUP BY LTRIM(m.value)
ORDER BY Registros DESC;

-- Evidencia: balances negativos aceptados (no son motivo de rechazo)
SELECT COUNT(*) AS AceptadosConBalanceNegativo   -- esperado: 30
FROM cur.PersonasValidas WHERE BalanceMensual < 0;
```

## Modelo relacional

El modelo entidad-relación se diseñó a partir de las columnas del Excel y está en tercera forma normal (Elmasri y Navathe, 2016). *Persona* agrupa los atributos que dependen solo del documento, y *Observacion* agrupa las respuestas de cada encuesta, que pueden cambiar en el tiempo. La Figura 6 presenta el diagrama.

**Figura 6**

*Modelo relacional de la base PersonasETL*



*Nota.* PK = clave primaria; FK = clave foránea; UQ = restricción de unicidad. En azul claro, los diez catálogos de categorías agregados en la corrección: TipoDocumento, Sexo, EstadoCivil, Zona, TipoVivienda, Estrato, TipoContrato, RegimenSalud, CondicionDiscapacidad y CanalOrigen.

**Ajustes al modelo para conservar la normalización**

- DepartmentName, MunicipalityName y OccupationName no se guardan en la observación, porque dependen de su código; se trasladaron a los catálogos Departamento, Municipio y Ocupacion.
- DepartmentCode tampoco se guarda en la observación, porque se deduce del municipio. Guardarlo habría permitido combinaciones contradictorias como las 22 detectadas.
- EmployerName se normalizó en la entidad Empleador, con clave sustituta.
- Persona y Observacion se separaron para admitir varias encuestas por persona. La clave de persona es (TipoDocumento, NumeroDocumento) y la de observación es (PersonaId, FechaEncuesta), ambas con restricción UNIQUE.
- Todas las categorías son tablas de catálogo con clave foránea: NivelEducativo y SituacionLaboral, que además alimentan dimensiones del DW, y los diez catálogos TipoDocumento, Sexo, EstadoCivil, Zona, TipoVivienda, Estrato, TipoContrato, RegimenSalud, CondicionDiscapacidad y CanalOrigen. Cada catálogo guarda el código y su nombre; la observación y la persona guardan solo el código, de modo que un valor que no exista en el catálogo no puede almacenarse.
- El código M999 ("MUNICIPIO DESCONOCIDO") no se incluyó en el catálogo porque no es una ubicación real; sus registros se envían a revisión. El código O99 ("SIN CLASIFICAR") sí se incluyó porque es un código válido de la fuente.
- Todas las columnas de texto son NVARCHAR, para coincidir con el tipo Unicode (DT_WSTR) que entrega SSIS desde Excel.

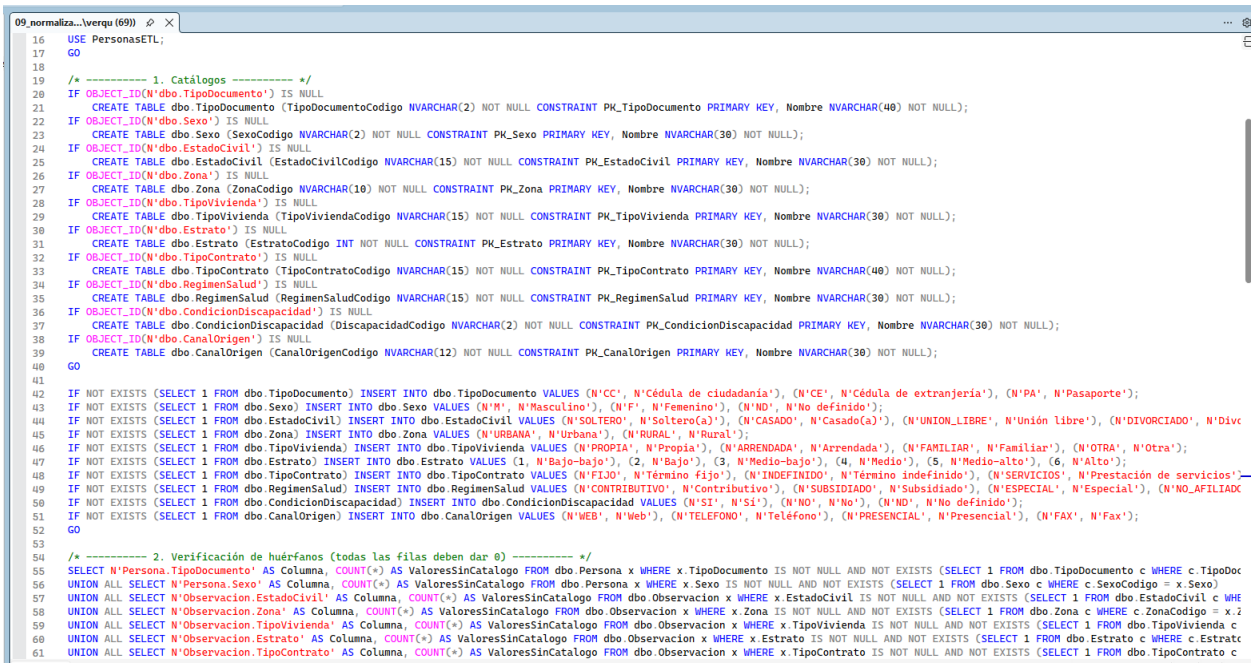
## Catálogos de categorías

En la primera versión, categorías como el tipo de vivienda, el estado civil o la zona se validaban con restricciones CHECK dentro de la tabla Observacion. Siguiendo la revisión del docente, cada una pasó a ser una tabla de catálogo referenciada con clave foránea, lo que completa la normalización: el significado de cada código se guarda una sola vez y se consulta con un JOIN. El script *04_modelo_relacional.sql* crea los catálogos para una instalación nueva, y el script *09_normalizacion_catalogos.sql* aplicó el mismo cambio a la base ya cargada sin borrar datos: crea y llena los catálogos, comprueba que ningún valor quede sin catálogo, quita las restricciones CHECK y crea las claves foráneas. Como los códigos guardados no cambian, los paquetes SSIS siguen funcionando igual.

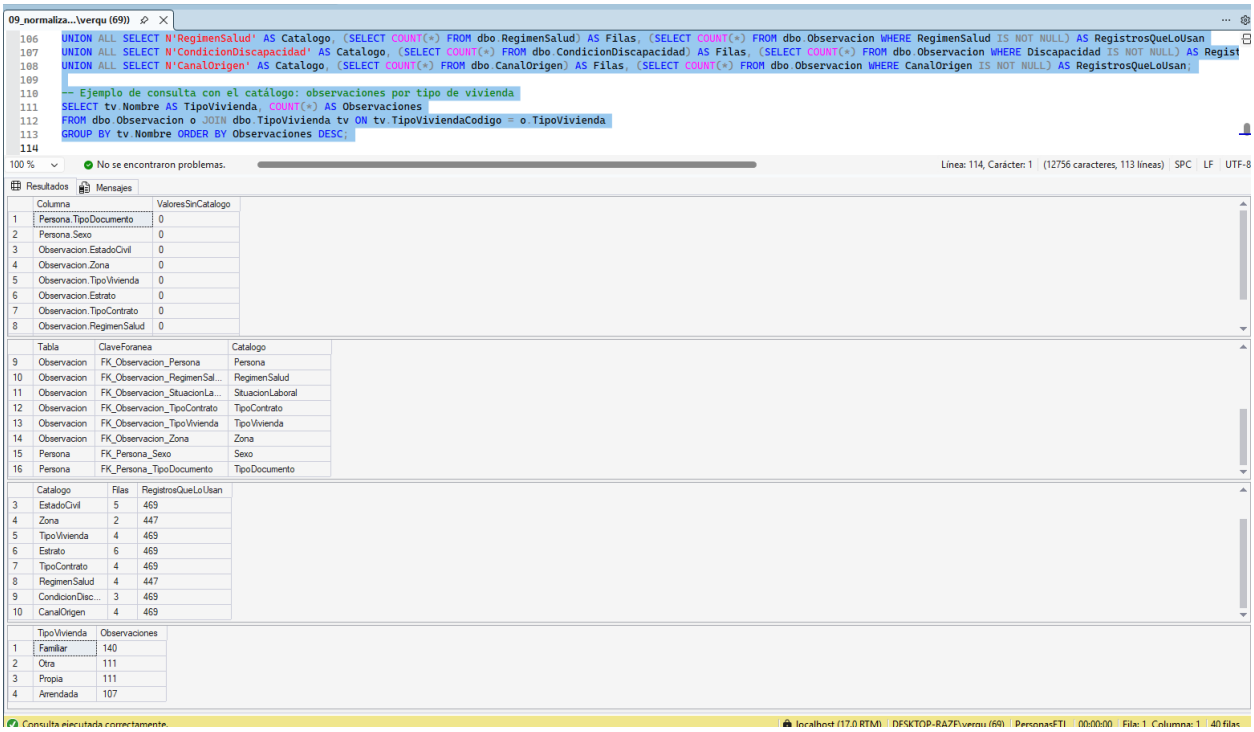
La Figura 7 muestra el script y su resultado: los diez catálogos tienen cero valores huérfanos, las claves foráneas nuevas aparecen en Persona y Observacion, y la consulta de ejemplo cuenta las observaciones por tipo de vivienda uniendo Observacion con su catálogo.

**Figura 7**

*Migración de las categorías a tablas de catálogo con el script 09*



```
09_normaliza...\verqu (69))
16  USE PersonasETL;
17  GO
18
19  /* ---------- 1. Catálogos ---------- */
20  IF OBJECT_ID(N'dbo.TipoDocumento') IS NULL
21      CREATE TABLE dbo.TipoDocumento (TipoDocumentoCodigo NVARCHAR(2) NOT NULL CONSTRAINT PK_TipoDocumento PRIMARY KEY, Nombre NVARCHAR(40) NOT NULL);
22  IF OBJECT_ID(N'dbo.Sexo') IS NULL
23      CREATE TABLE dbo.Sexo (SexoCodigo NVARCHAR(2) NOT NULL CONSTRAINT PK_Sexo PRIMARY KEY, Nombre NVARCHAR(30) NOT NULL);
24  IF OBJECT_ID(N'dbo.EstadoCivil') IS NULL
25      CREATE TABLE dbo.EstadoCivil (EstadoCivilCodigo NVARCHAR(15) NOT NULL CONSTRAINT PK_EstadoCivil PRIMARY KEY, Nombre NVARCHAR(30) NOT NULL);
26  IF OBJECT_ID(N'dbo.Zona') IS NULL
27      CREATE TABLE dbo.Zona (ZonaCodigo NVARCHAR(10) NOT NULL CONSTRAINT PK_Zona PRIMARY KEY, Nombre NVARCHAR(30) NOT NULL);
28  IF OBJECT_ID(N'dbo.TipoVivienda') IS NULL
29      CREATE TABLE dbo.TipoVivienda (TipoViviendaCodigo NVARCHAR(15) NOT NULL CONSTRAINT PK_TipoVivienda PRIMARY KEY, Nombre NVARCHAR(30) NOT NULL);
30  IF OBJECT_ID(N'dbo.Estrato') IS NULL
31      CREATE TABLE dbo.Estrato (EstratoCodigo INT NOT NULL CONSTRAINT PK_Estrato PRIMARY KEY, Nombre NVARCHAR(30) NOT NULL);
32  IF OBJECT_ID(N'dbo.TipoContrato') IS NULL
33      CREATE TABLE dbo.TipoContrato (TipoContratoCodigo NVARCHAR(15) NOT NULL CONSTRAINT PK_TipoContrato PRIMARY KEY, Nombre NVARCHAR(40) NOT NULL);
34  IF OBJECT_ID(N'dbo.RegimenSalud') IS NULL
35      CREATE TABLE dbo.RegimenSalud (RegimenSaludCodigo NVARCHAR(15) NOT NULL CONSTRAINT PK_RegimenSalud PRIMARY KEY, Nombre NVARCHAR(30) NOT NULL);
36  IF OBJECT_ID(N'dbo.CondicionDiscapacidad') IS NULL
37      CREATE TABLE dbo.CondicionDiscapacidad (DiscapacidadCodigo NVARCHAR(2) NOT NULL CONSTRAINT PK_CondicionDiscapacidad PRIMARY KEY, Nombre NVARCHAR(30) NOT NULL);
38  IF OBJECT_ID(N'dbo.CanalOrigen') IS NULL
39      CREATE TABLE dbo.CanalOrigen (CanalOrigenCodigo NVARCHAR(12) NOT NULL CONSTRAINT PK_CanalOrigen PRIMARY KEY, Nombre NVARCHAR(30) NOT NULL);
40  GO
41
42  IF NOT EXISTS (SELECT 1 FROM dbo.TipoDocumento) INSERT INTO dbo.TipoDocumento VALUES (N'CC', N'Cédula de ciudadanía'), (N'CE', N'Cédula de extranjería'), (N'PA', N'Pasaporte');
43  IF NOT EXISTS (SELECT 1 FROM dbo.Sexo) INSERT INTO dbo.Sexo VALUES (N'M', N'Masculino'), (N'F', N'Femenino'), (N'ND', N'No definido');
44  IF NOT EXISTS (SELECT 1 FROM dbo.EstadoCivil) INSERT INTO dbo.EstadoCivil VALUES (N'SOLTERO', N'Soltero(a)'), (N'CASADO', N'Casado(a)'), (N'UNION_LIBRE', N'Unión libre'), (N'DIVORCIADO', N'Divc
45  IF NOT EXISTS (SELECT 1 FROM dbo.Zona) INSERT INTO dbo.Zona VALUES (N'URBANA', N'Urbana'), (N'RURAL', N'Rural');
46  IF NOT EXISTS (SELECT 1 FROM dbo.TipoVivienda) INSERT INTO dbo.TipoVivienda VALUES (N'PROPIA', N'Propia'), (N'ARRENDADA', N'Arrendada'), (N'FAMILIAR', N'Familiar'), (N'OTRA', N'Otra');
47  IF NOT EXISTS (SELECT 1 FROM dbo.Estrato) INSERT INTO dbo.Estrato VALUES (1, N'Bajo-bajo'), (2, N'Bajo'), (3, N'Medio-bajo'), (4, N'Medio'), (5, N'Medio-alto'), (6, N'Alto');
48  IF NOT EXISTS (SELECT 1 FROM dbo.TipoContrato) INSERT INTO dbo.TipoContrato VALUES (N'FIJO', N'Término fijo'), (N'INDEFINIDO', N'Término indefinido'), (N'SERVICIOS', N'Prestación de servicios'),
49  IF NOT EXISTS (SELECT 1 FROM dbo.RegimenSalud) INSERT INTO dbo.RegimenSalud VALUES (N'CONTRIBUTIVO', N'Contributivo'), (N'SUBSIDIADO', N'Subsidiado'), (N'ESPECIAL', N'Especial'), (N'NO_AFILIADC
50  IF NOT EXISTS (SELECT 1 FROM dbo.CondicionDiscapacidad) INSERT INTO dbo.CondicionDiscapacidad VALUES (N'SI', N'Sí'), (N'NO', N'No'), (N'ND', N'No definido');
51  IF NOT EXISTS (SELECT 1 FROM dbo.CanalOrigen) INSERT INTO dbo.CanalOrigen VALUES (N'WEB', N'Web'), (N'TELEFONO', N'Teléfono'), (N'PRESENCIAL', N'Presencial'), (N'FAX', N'Fax');
52  GO
53
54  /* ---------- 2. Verificación de huérfanos (todas las filas deben dar 0) ---------- */
55  SELECT N'Persona.TipoDocumento' AS Columna, COUNT(*) AS ValoresSinCatalogo FROM dbo.Persona x WHERE x.TipoDocumento IS NOT NULL AND NOT EXISTS (SELECT 1 FROM dbo.TipoDocumento c WHERE c.TipoDoc
56  UNION ALL SELECT N'Persona.Sexo' AS Columna, COUNT(*) AS ValoresSinCatalogo FROM dbo.Persona x WHERE x.Sexo IS NOT NULL AND NOT EXISTS (SELECT 1 FROM dbo.Sexo c WHERE c.SexoCodigo = x.Sexo)
57  UNION ALL SELECT N'Observacion.EstadoCivil' AS Columna, COUNT(*) AS ValoresSinCatalogo FROM dbo.Observacion x WHERE x.EstadoCivil IS NOT NULL AND NOT EXISTS (SELECT 1 FROM dbo.EstadoCivil c WHE
58  UNION ALL SELECT N'Observacion.Zona' AS Columna, COUNT(*) AS ValoresSinCatalogo FROM dbo.Observacion x WHERE x.Zona IS NOT NULL AND NOT EXISTS (SELECT 1 FROM dbo.Zona c WHERE c.ZonaCodigo = x.Z
59  UNION ALL SELECT N'Observacion.TipoVivienda' AS Columna, COUNT(*) AS ValoresSinCatalogo FROM dbo.Observacion x WHERE x.TipoVivienda IS NOT NULL AND NOT EXISTS (SELECT 1 FROM dbo.TipoVivienda c
60  UNION ALL SELECT N'Observacion.Estrato' AS Columna, COUNT(*) AS ValoresSinCatalogo FROM dbo.Observacion x WHERE x.Estrato IS NOT NULL AND NOT EXISTS (SELECT 1 FROM dbo.Estrato c WHERE c.Estratc
61  UNION ALL SELECT N'Observacion.TipoContrato' AS Columna, COUNT(*) AS ValoresSinCatalogo FROM dbo.Observacion x WHERE x.TipoContrato IS NOT NULL AND NOT EXISTS (SELECT 1 FROM dbo.TipoContrato c
```



```
UNION ALL SELECT N'RegimenSalud' AS Catalogo, (SELECT COUNT(*) FROM dbo.RegimenSalud) AS Filas, (SELECT COUNT(*) FROM dbo.Observacion WHERE RegimenSalud IS NOT NULL) AS RegistrosQueLoUsan
UNION ALL SELECT N'CondicionDiscapacidad' AS Catalogo, (SELECT COUNT(*) FROM dbo.CondicionDiscapacidad) AS Filas, (SELECT COUNT(*) FROM dbo.Observacion WHERE Discapacidad IS NOT NULL) AS Regist
UNION ALL SELECT N'CanalOrigen' AS Catalogo, (SELECT COUNT(*) FROM dbo.CanalOrigen) AS Filas, (SELECT COUNT(*) FROM dbo.Observacion WHERE CanalOrigen IS NOT NULL) AS RegistrosQueLoUsan;

-- Ejemplo de consulta con el catálogo: observaciones por tipo de vivienda
SELECT tv.Nombre AS TipoVivienda, COUNT(*) AS Observaciones
FROM dbo.Observacion o JOIN dbo.TipoVivienda tv ON tv.TipoViviendaCodigo = o.TipoVivienda
GROUP BY tv.Nombre ORDER BY Observaciones DESC;
```

| | Columna | ValoresSinCatalogo |
|---|---|---|
| 1 | Persona.TipoDocumento | 0 |
| 2 | Persona.Sexo | 0 |
| 3 | Observacion.EstadoCivil | 0 |
| 4 | Observacion.Zona | 0 |
| 5 | Observacion.TipoVivienda | 0 |
| 6 | Observacion.Estrato | 0 |
| 7 | Observacion.TipoContrato | 0 |
| 8 | Observacion.RegimenSalud | 0 |

| | Tabla | ClaveForanea | Catalogo |
|---|---|---|---|
| 9 | Observacion | FK_Observacion_Persona | Persona |
| 10 | Observacion | FK_Observacion_RegimenSal... | RegimenSalud |
| 11 | Observacion | FK_Observacion_SituacionLa... | SituacionLaboral |
| 12 | Observacion | FK_Observacion_TipoContrato | TipoContrato |
| 13 | Observacion | FK_Observacion_TipoVivienda | TipoVivienda |
| 14 | Observacion | FK_Observacion_Zona | Zona |
| 15 | Persona | FK_Persona_Sexo | Sexo |
| 16 | Persona | FK_Persona_TipoDocumento | TipoDocumento |

| | Catalogo | Filas | RegistrosQueLoUsan |
|---|---|---|---|
| 3 | EstadoCivil | 5 | 469 |
| 4 | Zona | 2 | 447 |
| 5 | TipoVivienda | 4 | 469 |
| 6 | Estrato | 6 | 469 |
| 7 | TipoContrato | 4 | 469 |
| 8 | RegimenSalud | 4 | 447 |
| 9 | CondicionDisc... | 3 | 469 |
| 10 | CanalOrigen | 4 | 469 |

| | TipoVivienda | Observaciones |
|---|---|---|
| 1 | Familiar | 140 |
| 2 | Otra | 111 |
| 3 | Propia | 111 |
| 4 | Arrendada | 107 |

*Nota.* Arriba, la creación y el llenado de los catálogos. Abajo, la verificación de huérfanos (todas en 0), las claves foráneas, las filas de cada catálogo y las observaciones por tipo de vivienda: Familiar 140, Otra 111, Propia 111 y Arrendada 107, que suman 469.

## Correspondencia entre el Excel y el modelo relacional

La Tabla 5 relaciona cada columna del archivo de origen con la entidad y el atributo de destino, junto con el tratamiento aplicado.

**Tabla 5**

*Correspondencia entre las columnas del Excel y el modelo relacional*

| Columna Excel | Entidad.atributo | Tratamiento |
|---|---|---|
| SourceRowId | Observacion.SourceRowId | Entero; trazabilidad |
| DocumentType | Persona.TipoDocumento | Mayúsculas; FK al catálogo TipoDocumento |
| DocumentNumber | Persona.NumeroDocumento | Mayúsculas; SIM + 7 dígitos; obligatorio |
| FirstName / MiddleName | Persona.PrimerNombre / SegundoNombre | Nombre propio; primer nombre obligatorio |
| LastName / SecondLastName | Persona.PrimerApellido / SegundoApellido | Nombre propio; primer apellido obligatorio |
| BirthDate | Persona.FechaNacimiento | dd/mm/aaaa a ISO; DATE; no futura |
| ReportedAge | Observacion.EdadReportada | Entero; coherente con las fechas |
| Sex | Persona.Sexo | M, F o ND; otro valor a NULL; FK a Sexo |
| MaritalStatus | Observacion.EstadoCivil | Mayúsculas; FK al catálogo EstadoCivil |
| Email | Persona.Email | Minúsculas; con @ y dominio; si no, NULL |
| Phone | Persona.Telefono | Sin guiones; 10 dígitos; si no, NULL |
| DepartmentCode / DepartmentName | Departamento (catálogo) | Validación de coherencia |
| MunicipalityCode | Observacion.MunicipioCodigo | Lookup al catálogo Municipio |
| MunicipalityName | Municipio.Nombre (catálogo) | Nombre oficial del catálogo |
| Zone | Observacion.Zona | URBANA o RURAL; otro valor a NULL; FK a Zona |
| Address | Observacion.Direccion | TRIM |
| HousingType | Observacion.TipoVivienda | Mayúsculas; FK al catálogo TipoVivienda |
| SocioeconomicStratum | Observacion.Estrato | Entero 1 a 6; otro valor a NULL; FK a Estrato |
| EducationLevel | Observacion.NivelEducativoCodigo | Mayúsculas; Lookup |
| EmploymentStatus | Observacion.SituacionLaboralCodigo | Mayúsculas; Lookup |
| OccupationCode | Observacion.OcupacionCodigo | Lookup; NULL si no aplica |
| OccupationName | Ocupacion.Nombre (catálogo) | No se almacena en la observación |
| EmployerName | Empleador.Nombre / Observacion.EmpleadorId | Entidad propia; clave sustituta |
| ContractType | Observacion.TipoContrato | Mayúsculas; FK a TipoContrato; coherencia laboral |
| EmploymentStartDate | Observacion.FechaInicioEmpleo | DATE; entre nacimiento y encuesta |

| Columna Excel | Entidad.atributo | Tratamiento |
|---|---|---|
| MonthlyIncome / MonthlyExpenses | Observacion.IngresoMensual / GastoMensual | Limpieza monetaria; DECIMAL(18,2); mayor o igual a 0 |
| Dependents | Observacion.PersonasACargo | Entero mayor o igual a 0 |
| HouseholdSize | Observacion.TamanoHogar | Entero mayor o igual a 1 |
| HealthRegime | Observacion.RegimenSalud | "NULL", "sin dato" y "N/A" a NULL; FK a RegimenSalud |
| Disability | Observacion.Discapacidad | sí o 1 a SI; no o 0 a NO; FK a CondicionDiscapacidad |
| SurveyDate | Observacion.FechaEncuesta | DATE; obligatoria |
| UpdatedAt | Observacion.FechaActualizacionOrigen | DATETIME; desempata duplicados |
| SourceChannel | Observacion.CanalOrigen | Mayúsculas; FK al catálogo CanalOrigen |

## Reglas de validación y tratamiento

### Destinos de los registros

Cada registro termina en uno de cuatro destinos según la gravedad del problema (Tabla 6). Esta separación permite aplicar reglas estrictas sin inventar datos: lo que no se puede identificar se rechaza, lo identificable pero dudoso se aparta para revisión y lo secundario se corrige o se deja en NULL con registro del cambio.

**Tabla 6**

*Destinos posibles de un registro*

| Destino | Criterio | Tabla |
|---|---|---|
| Rechazado | No se puede identificar la persona o la observación: sin documento válido, sin tipo de documento válido, sin primer nombre o apellido, o sin fecha de encuesta válida. | etl.Rechazo |
| Duplicado descartado | Otra versión de la misma observación (persona y fecha) que no es la vigente. | etl.Duplicado |
| En revisión | Identificable, pero con datos analíticos inconsistentes o desconocidos (fechas, edad, ubicación, importes, hogar o información laboral). No se carga y conserva todos sus motivos. | etl.Revision |
| Válido | Supera todas las reglas. Los atributos secundarios inválidos quedan en NULL y el cambio se registra en etl.LogCambio. | stg.PersonaDepurada |

## Reglas aplicadas

La Tabla 7 detalla cómo se implementó cada regla exigida y qué componente de SSIS la ejecuta.

**Tabla 7**

*Reglas de validación y tratamiento implementadas*

| Regla | Implementación | Componente | Resultado |
|---|---|---|---|
| Normalización de textos | TRIM; mayúsculas en códigos y categorías; nombre propio en nombres; minúsculas en email | DC1 | Corrección registrada |
| Conversión de tipos | Texto a DATE, DATETIME, DECIMAL(18,2) e INT con error ignorado: la falla deja NULL y una regla la detecta | Data Conversion, DC2 | NULL detectable |
| Valores faltantes | Claves faltantes, a rechazo; importe faltante, a revisión; atributo secundario, a NULL | DC3, DC4 | Rechazo, revisión o NULL |
| Documentos y contacto | Documento SIM + 7 dígitos y tipo CC, CE o PA; email con @ y dominio; teléfono de 10 dígitos | DC1 a DC3 | Rechazo o NULL |
| Fechas | Fechas inexistentes, nacimiento futuro, actualización anterior a la encuesta, inicio de empleo fuera de rango | DC2, DC4 | Rechazo o revisión |
| Edad, nacimiento y encuesta | Diferencia mayor que un año entre la edad calculada y la reportada | DC4 | Revisión |
| Municipio y departamento | El municipio existe en el catálogo y su departamento coincide con el código y el nombre informados | Lookups, DC4 | Revisión |
| Categorías | Mayúsculas, Lookup a catálogos y clave foránea a las tablas de catálogo | DC1, Lookups, DC4 | Revisión si no existe |
| Ingresos y gastos | Limpieza de símbolos y separadores; numérico; mayor o igual a 0 | DC1, DC2, DC4 | Revisión |
| Personas a cargo y hogar | Enteros válidos (cargo ≥ 0, hogar ≥ 1); si cargo > hogar se registra advertencia | DC2 a DC4 | Revisión o advertencia |
| Información laboral | EMPLEADO con contrato y ocupación; INDEPENDIENTE con ocupación y sin empleador; los demás sin datos laborales | DC4 | Revisión |
| Duplicados | Misma persona y SurveyDate; se conserva el UpdatedAt más reciente y, si empata, el menor SourceRowId | Aggregate, Sort, Merge Join, Conditional Split | Descartado y registrado |

**Resolución de duplicados y justificación**

Siguiendo el enunciado, la clave de persona es el tipo y número de documento, y la clave de observación es la persona junto con la fecha de encuesta. El diagnóstico encontró 50 pares con la misma clave de observación. En 25 pares los registros son idénticos; en los otros 25 solo cambian MonthlyIncome (500.000 más) y UpdatedAt (de 12:30 a 18:45 del mismo día). Se conserva la versión con UpdatedAt más reciente, porque refleja la última actualización del sistema de origen. Si UpdatedAt empata, se conserva la primera aparición (menor SourceRowId); como el contenido es igual, la elección solo afecta la trazabilidad. Así, entre las filas 826 y 976 se conserva la 976, con ingreso de 15.200.000, y entre las filas 801 y 951 se conserva la 801.

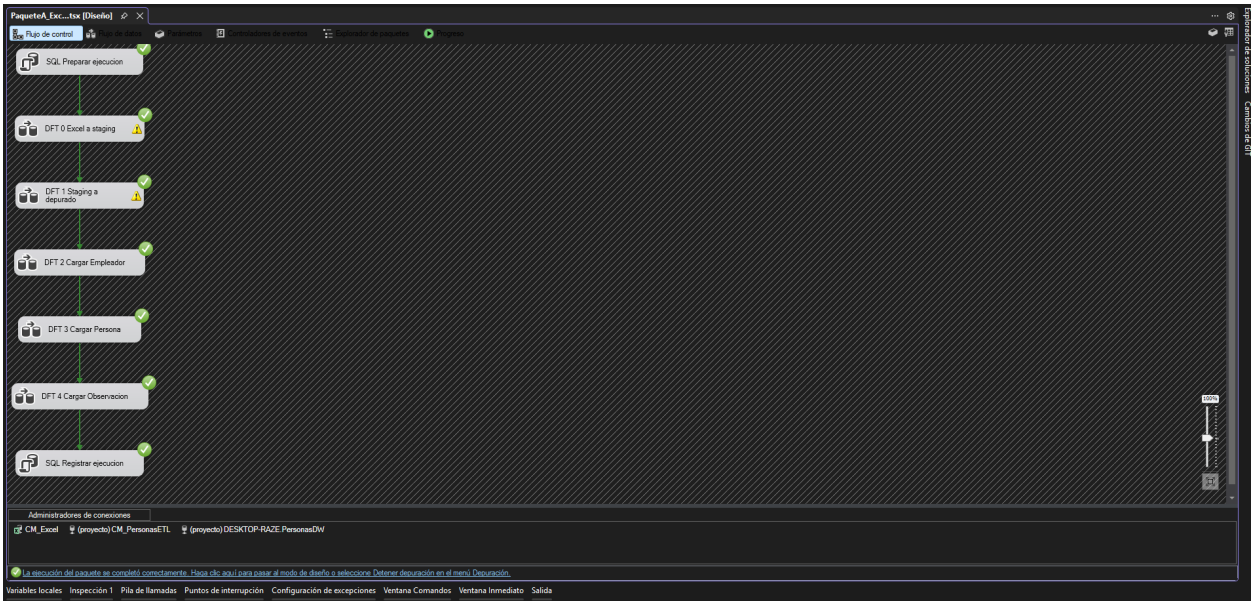
En SSIS se implementó sin Script Component. Una columna numérica RankKey, formada por la fecha de actualización (AAAAMMDDhhmmss) multiplicada por 10.000 más 9.999 menos el SourceRowId, ordena las versiones. Aggregate calcula el máximo por clave, Merge Join devuelve ese máximo a cada fila y Conditional Split separa la versión ganadora de las descartadas.

## Paquete SSIS A: Excel hacia el modelo relacional

El paquete A tiene siete tareas en su flujo de control (Figura 8) y trabaja en dos pasos explícitos. Primero, el DFT 0 copia el Excel sin ningún cambio a la tabla de staging *stg.PersonasRaw*. Después, el DFT 1 lee ese staging y concentra la limpieza, la validación y la deduplicación. Las tres tareas siguientes cargan el modelo relacional respetando las claves primarias y foráneas.

**Figura 8**

*Flujo de control del paquete A ejecutado*



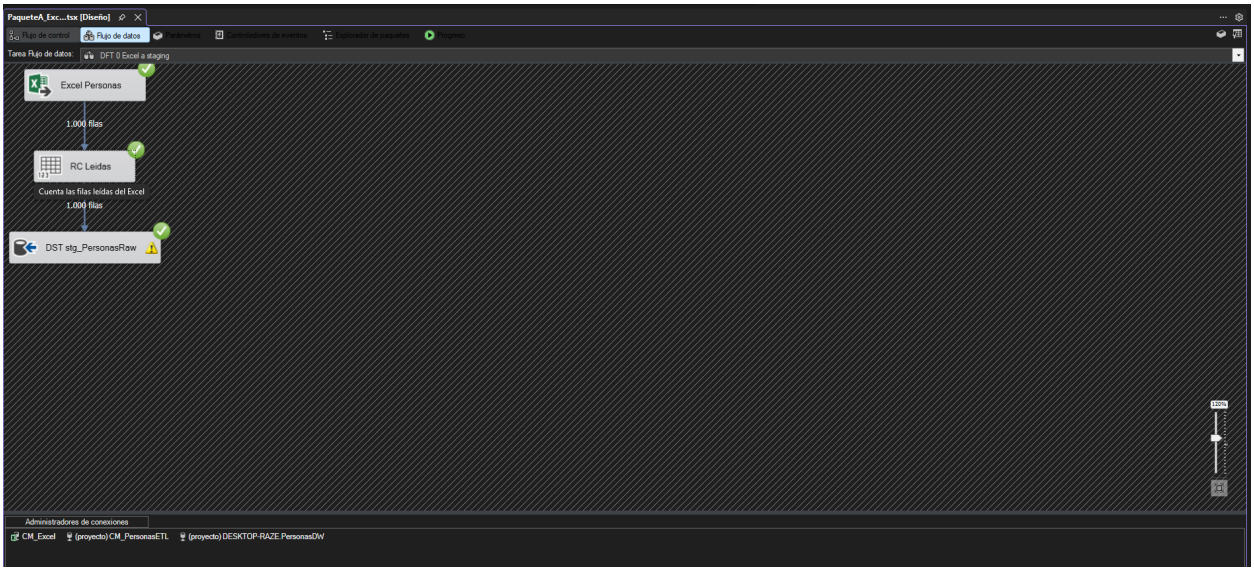

*Nota.* Las siete tareas terminaron correctamente. El triángulo amarillo es una advertencia de posible truncamiento de texto que no afecta los datos.

## Carga del Excel al staging

El DFT 0 tiene tres componentes: el origen de Excel, un contador de filas y el destino OLE DB hacia *stg.PersonasRaw* (Figura 9). No transforma nada: las 37 columnas se guardan como texto, exactamente como están en el archivo, para que el staging sea la evidencia del estado original.

**Figura 9**

*Ejecución del flujo de datos DFT 0 Excel a staging*

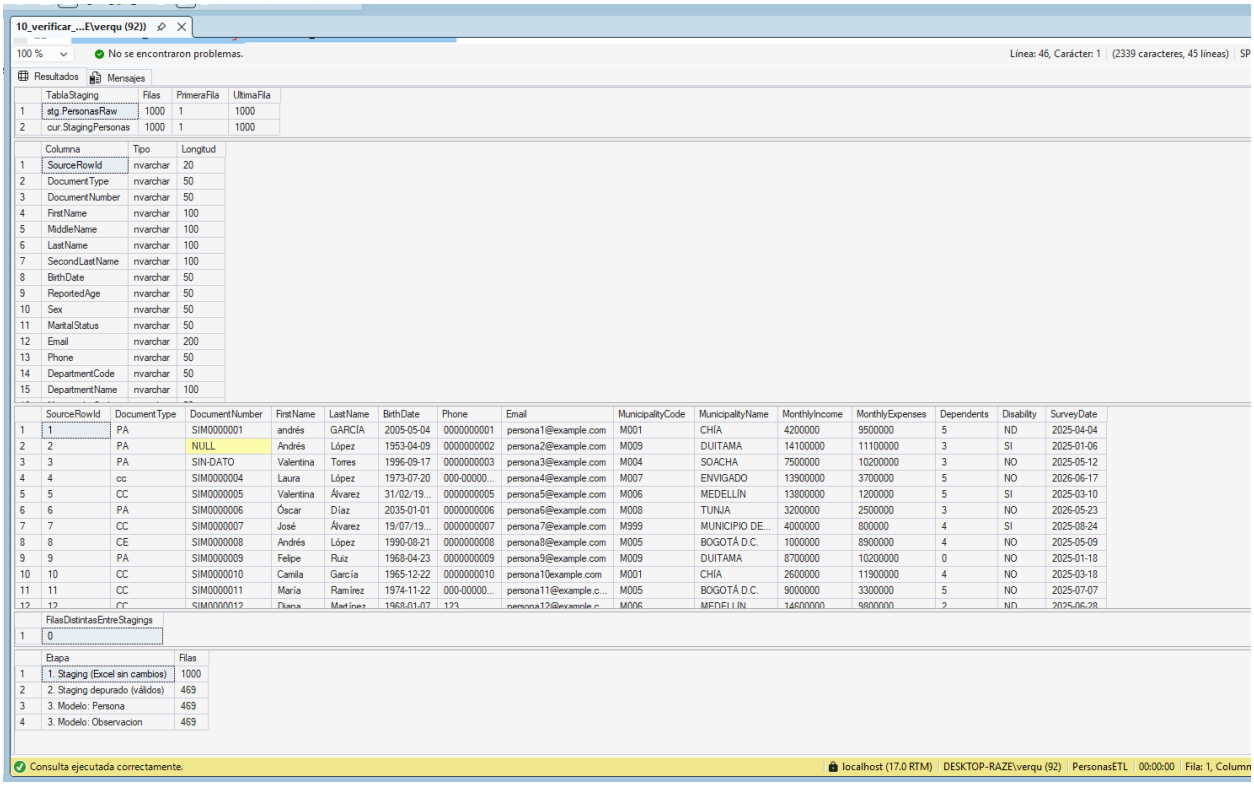



*Nota.* Las 1.000 filas del Excel pasan por el contador y llegan a stg.PersonasRaw.

La tabla del staging se verificó en SQL Server con el script *10_verificar_staging.sql* (Figura 10). La tabla tiene 1.000 filas (SourceRowId 1 a 1.000), 37 columnas NVARCHAR y conserva los valores sin limpiar, por ejemplo "cc" en la fila 4, el documento vacío de la fila 2 o "31/02/19..." en la fila 5. Es idéntica a la copia usada por el cursor (0 filas distintas), y el recorrido del dato muestra 1.000 filas en el staging, 469 en el staging depurado y 469 personas y observaciones en el modelo.

**Figura 10**

*Verificación en SQL Server de la tabla de staging stg.PersonasRaw*



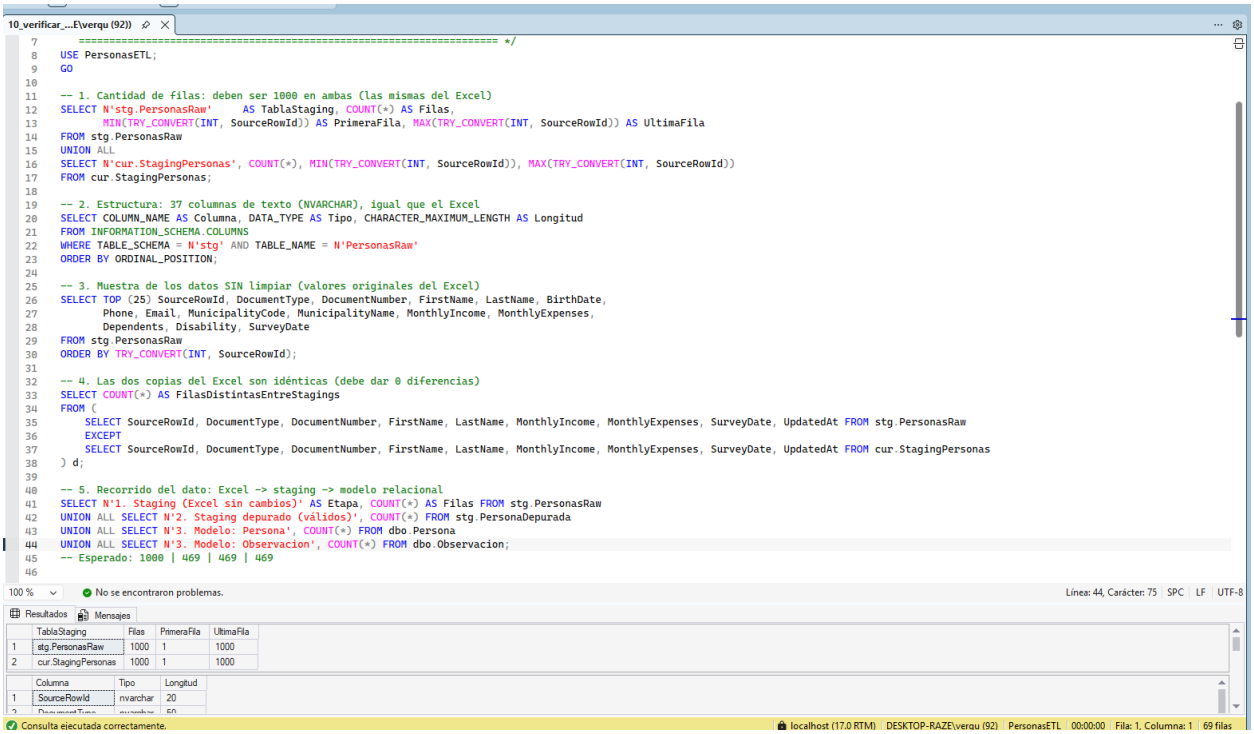
| | TablaStaging | Filas | PrimeraFila | UltimaFila |
|---|---|---|---|---|
| 1 | stg.PersonasRaw | 1000 | 1 | 1000 |
| 2 | cur.StagingPersonas | 1000 | 1 | 1000 |

| | Columna | Tipo | Longitud |
|---|---|---|---|
| 1 | SourceRowId | nvarchar | 20 |
| 2 | DocumentType | nvarchar | 50 |
| 3 | DocumentNumber | nvarchar | 50 |
| 4 | FirstName | nvarchar | 100 |
| 5 | MiddleName | nvarchar | 100 |
| 6 | LastName | nvarchar | 100 |
| 7 | SecondLastName | nvarchar | 100 |
| 8 | BirthDate | nvarchar | 50 |
| 9 | ReportedAge | nvarchar | 50 |
| 10 | Sex | nvarchar | 50 |
| 11 | MaritalStatus | nvarchar | 50 |
| 12 | Email | nvarchar | 200 |
| 13 | Phone | nvarchar | 50 |
| 14 | DepartmentCode | nvarchar | 50 |
| 15 | DepartmentName | nvarchar | 100 |

| | SourceRowId | DocumentType | DocumentNumber | FirstName | LastName | BirthDate | Phone | Email | MunicipalityCode | MunicipalityName | MonthlyIncome | MonthlyExpenses | Dependents | Disability | SurveyDate |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | PA | SIM0000001 | andrés | GARCÍA | 2005-05-04 | 0000000001 | persona1@example.com | M001 | CHÍA | 4200000 | 9500000 | 5 | ND | 2025-04-04 |
| 2 | 2 | PA | NULL | Andrés | López | 1953-04-09 | 0000000002 | persona2@example.com | M009 | DUITAMA | 14100000 | 11100000 | 3 | SI | 2025-01-06 |
| 3 | 3 | PA | SIN-DATO | Valentina | Torres | 1996-09-17 | 0000000003 | persona3@example.com | M004 | SOACHA | 7500000 | 10200000 | 3 | NO | 2025-05-12 |
| 4 | 4 | cc | SIM0000004 | Laura | López | 1973-07-20 | 000-00000... | persona4@example.com | M007 | ENVIGADO | 13900000 | 3700000 | 5 | NO | 2026-06-17 |
| 5 | 5 | CC | SIM0000005 | Valentina | Álvarez | 31/02/19... | 0000000005 | persona5@example.com | M006 | MEDELLÍN | 13800000 | 1200000 | 5 | SI | 2025-03-10 |
| 6 | 6 | PA | SIM0000006 | Óscar | Díaz | 2035-01-01 | 0000000006 | persona6@example.com | M008 | TUNJA | 3200000 | 2500000 | 3 | NO | 2026-05-23 |
| 7 | 7 | CC | SIM0000007 | José | Álvarez | 19/07/19... | 0000000007 | persona7@example.com | M999 | MUNICIPIO DE... | 4000000 | 800000 | 4 | SI | 2025-08-24 |
| 8 | 8 | CE | SIM0000008 | Andrés | López | 1990-08-21 | 0000000008 | persona8@example.com | M005 | BOGOTÁ D.C. | 1000000 | 8900000 | 4 | NO | 2025-05-09 |
| 9 | 9 | PA | SIM0000009 | Felipe | Ruiz | 1968-04-23 | 0000000009 | persona9@example.com | M009 | DUITAMA | 8700000 | 10200000 | 0 | NO | 2025-01-18 |
| 10 | 10 | CC | SIM0000010 | Camila | García | 1965-12-22 | 0000000010 | persona10example.com | M001 | CHÍA | 2600000 | 11900000 | 4 | NO | 2025-03-18 |
| 11 | 11 | CC | SIM0000011 | María | Ramírez | 1974-11-22 | 000-00000... | persona11@example.c... | M005 | BOGOTÁ D.C. | 9000000 | 3300000 | 5 | NO | 2025-07-07 |
| 12 | 12 | CC | SIM0000012 | Diana | Martínez | 1968-01-07 | 123 | persona12@example.c... | M006 | MEDELLÍN | 14600000 | 9800000 | 2 | ND | 2025-06-28 |

| | FilasDistintasEntreStagings |
|---|---|
| 1 | 0 |

| | Etapa | Filas |
|---|---|---|
| 1 | 1. Staging (Excel sin cambios) | 1000 |
| 2 | 2. Staging depurado (válidos) | 469 |
| 3 | 3. Modelo: Persona | 469 |
| 4 | 3. Modelo: Observacion | 469 |



```
   ================================================================ */
USE PersonasETL;
GO

-- 1. Cantidad de filas: deben ser 1000 en ambas (las mismas del Excel)
SELECT N'stg.PersonasRaw'     AS TablaStaging, COUNT(*) AS Filas,
       MIN(TRY_CONVERT(INT, SourceRowId)) AS PrimeraFila, MAX(TRY_CONVERT(INT, SourceRowId)) AS UltimaFila
FROM stg.PersonasRaw
UNION ALL
SELECT N'cur.StagingPersonas', COUNT(*), MIN(TRY_CONVERT(INT, SourceRowId)), MAX(TRY_CONVERT(INT, SourceRowId))
FROM cur.StagingPersonas;

-- 2. Estructura: 37 columnas de texto (NVARCHAR), igual que el Excel
SELECT COLUMN_NAME AS Columna, DATA_TYPE AS Tipo, CHARACTER_MAXIMUM_LENGTH AS Longitud
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_SCHEMA = N'stg' AND TABLE_NAME = N'PersonasRaw'
ORDER BY ORDINAL_POSITION;

-- 3. Muestra de los datos SIN limpiar (valores originales del Excel)
SELECT TOP (25) SourceRowId, DocumentType, DocumentNumber, FirstName, LastName, BirthDate,
       Phone, Email, MunicipalityCode, MunicipalityName, MonthlyIncome, MonthlyExpenses,
       Dependents, Disability, SurveyDate
FROM stg.PersonasRaw
ORDER BY TRY_CONVERT(INT, SourceRowId);

-- 4. Las dos copias del Excel son idénticas (debe dar 0 diferencias)
SELECT COUNT(*) AS FilasDistintasEntreStagings
FROM (
    SELECT SourceRowId, DocumentType, DocumentNumber, FirstName, LastName, MonthlyIncome, MonthlyExpenses, SurveyDate, UpdatedAt FROM stg.PersonasRaw
    EXCEPT
    SELECT SourceRowId, DocumentType, DocumentNumber, FirstName, LastName, MonthlyIncome, MonthlyExpenses, SurveyDate, UpdatedAt FROM cur.StagingPersonas
) d;

-- 5. Recorrido del dato: Excel -> staging -> modelo relacional
SELECT N'1. Staging (Excel sin cambios)' AS Etapa, COUNT(*) AS Filas FROM stg.PersonasRaw
UNION ALL SELECT N'2. Staging depurado (válidos)', COUNT(*) FROM stg.PersonaDepurada
UNION ALL SELECT N'3. Modelo: Persona', COUNT(*) FROM dbo.Persona
UNION ALL SELECT N'3. Modelo: Observacion', COUNT(*) FROM dbo.Observacion;
-- Esperado: 1000 | 469 | 469 | 469
```

| | TablaStaging | Filas | PrimeraFila | UltimaFila |
|---|---|---|---|---|
| 1 | stg.PersonasRaw | 1000 | 1 | 1000 |
| 2 | cur.StagingPersonas | 1000 | 1 | 1000 |

| | Columna | Tipo | Longitud |
|---|---|---|---|
| 1 | SourceRowId | nvarchar | 20 |

*Nota.* Arriba, los resultados: conteo, estructura, muestra de datos originales, comparación entre stagings y recorrido del dato. Abajo, el script.

## Del staging al modelo relacional

La Tabla 8 describe cada componente. La Figura 11 esquematiza los dos flujos y la Figura 12 muestra la ejecución del DFT 1 con el número de filas que pasó por cada flecha. No se utilizó Script Component: toda la transformación se hizo con componentes nativos, y la única lógica SQL dentro del paquete es la preparación de las tablas de trabajo y el registro de contadores, que no transforman datos.

**Tabla 8**

*Componentes del paquete A*

| Componente | Tipo | Función |
|---|---|---|
| SQL Preparar ejecución | Execute SQL Task | Vacía las tablas de staging y bitácora de la corrida anterior; el modelo nunca se vacía |
| DFT 0: Excel Personas | Excel Source | Lee la hoja Personas$ sin modificar el archivo |
| DFT 0: RC Leidas | Row Count | Cuenta las filas leídas del Excel |
| DFT 0: DST stg_PersonasRaw | OLE DB Destination | Guarda las 1.000 filas tal como llegaron en stg.PersonasRaw |
| DFT 1: SRC stg_PersonasRaw | OLE DB Source | Lee del staging las 37 columnas originales; aquí empieza la transformación |
| DC1 Normalizar texto | Derived Column | Normaliza textos y categorías y prepara fechas e importes |
| DCV Enteros | Data Conversion | Convierte SourceRowId, ReportedAge y HouseholdSize a entero, con error ignorado |
| DC2 Convertir tipos y claves | Derived Column | Convierte fechas, importes y enteros; calcula ClaveObservacion y RankKey |
| DC3 Rechazo y cambios | Derived Column | Construye el motivo de rechazo y la lista de correcciones |
| CS Con cambios y DST etl_LogCambio | Conditional Split y destino | Registra las correcciones por fila |
| CS Rechazo, RC Rechazadas y DST etl_Rechazo | Conditional Split, Row Count y destino | Separa los registros rechazados con su motivo |
| MC Dedup, AGG, SRT, MJ y CS Duplicado | Multicast, Aggregate, Sort, Merge Join, Conditional Split | Resuelven los duplicados y los registran en etl.Duplicado |
| LK Municipio, Departamento, NivelEducativo, SituacionLaboral y Ocupacion | Lookup | Validan contra los catálogos |
| DC4 y DC5 | Derived Column | Aplican las reglas de revisión y construyen el motivo |
| CS Revision, RC y destinos | Conditional Split, Row Count y destinos | Envían cada fila a etl.Revision o a stg.PersonaDepurada |
| DFT 2 Cargar Empleador | Aggregate y Lookup | Inserta solo los empleadores nuevos |
| DFT 3 Cargar Persona | Sort (sin duplicados) y Lookup | Inserta solo las personas nuevas |
| DFT 4 Cargar Observacion | Lookup | Resuelve PersonaId y EmpleadorId e inserta solo las observaciones nuevas |
| SQL Registrar ejecución | Execute SQL Task | Guarda los contadores en etl.EjecucionPaquete |

**Figura 11**

*Esquema de los flujos DFT 0 Excel a staging y DFT 1 Staging a depurado*

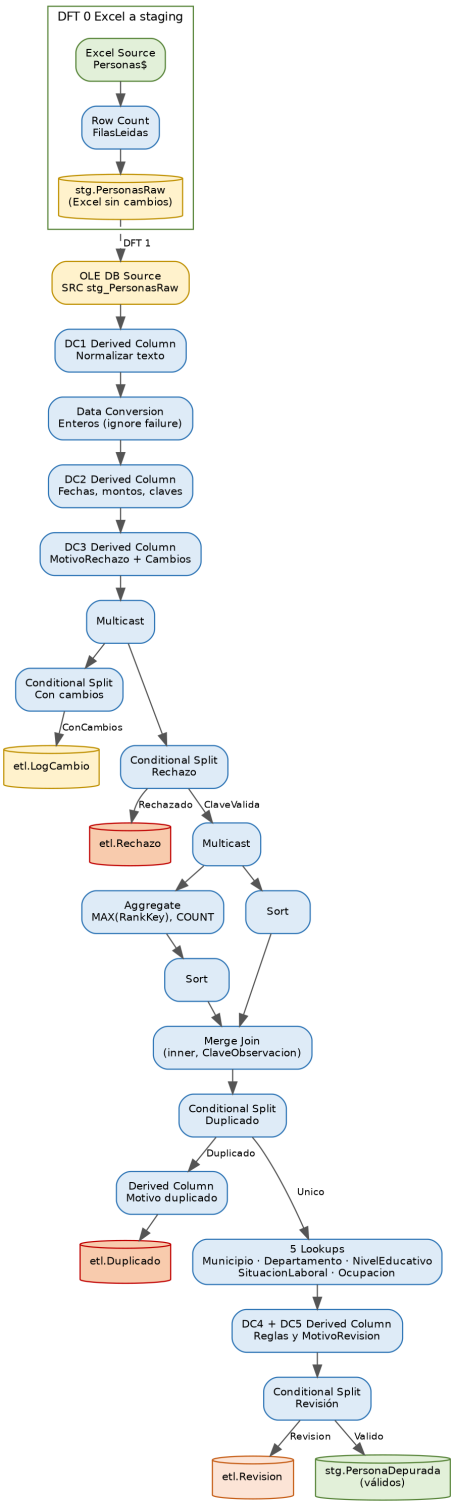


**Figura 12**

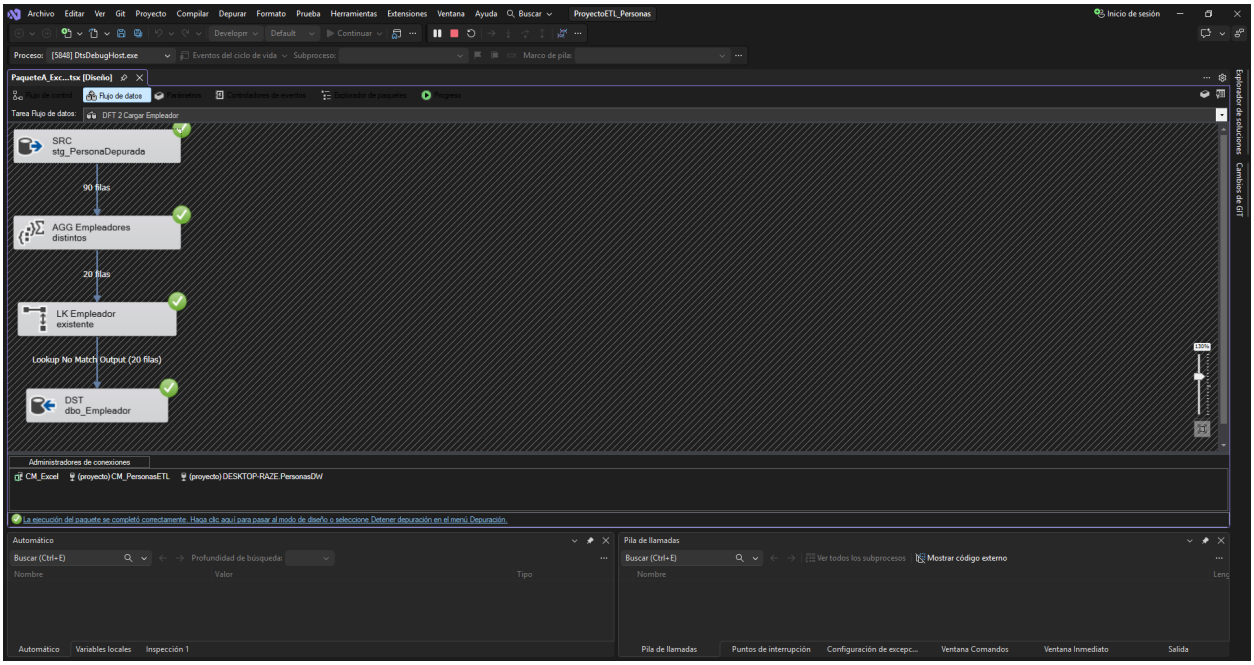
*Ejecución del flujo de datos DFT 1 Staging a depurado*

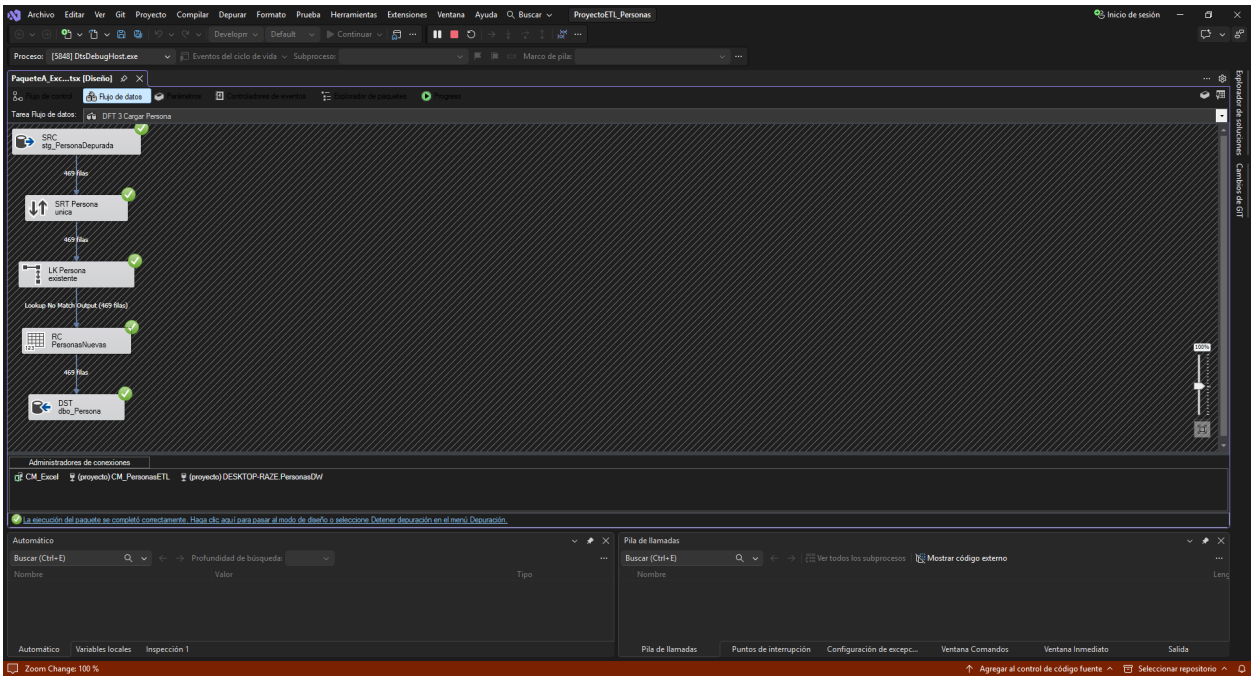
*Nota.* De izquierda a derecha, el flujo de arriba abajo: 1.000 filas leídas del staging; 90 rechazadas y 910 con clave válida; 509 filas con correcciones; 50 duplicadas y 860 únicas; 391 enviadas a revisión y 469 válidas.

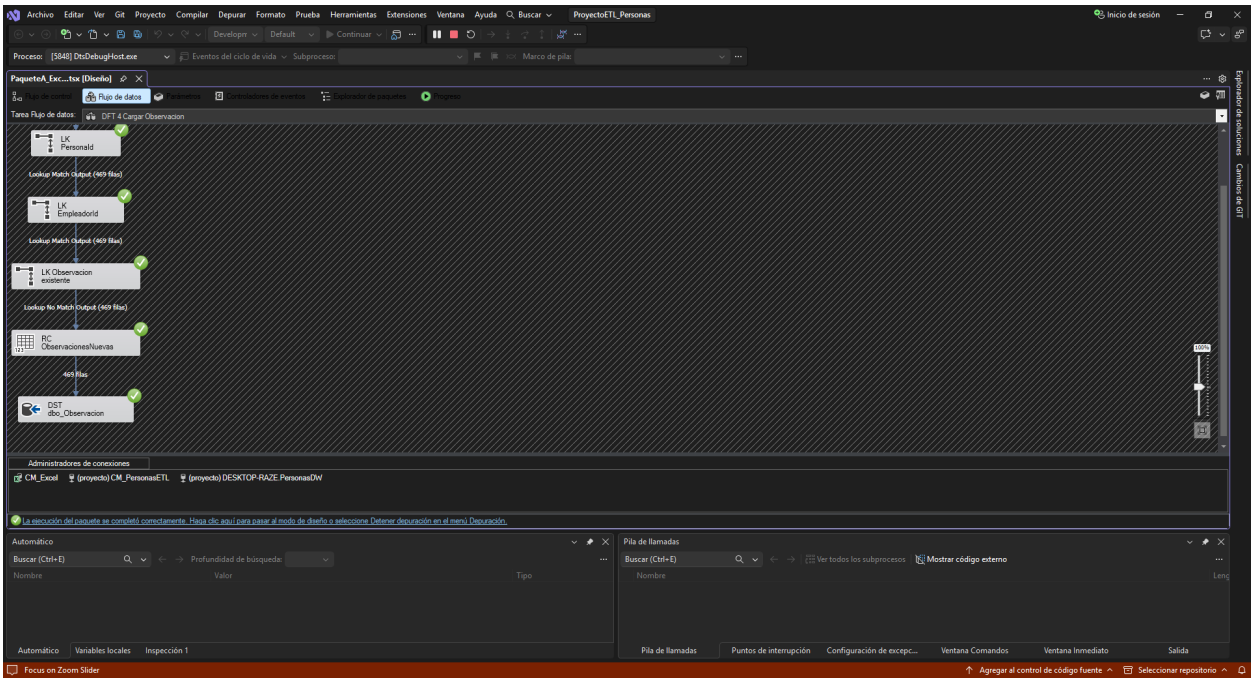
Los flujos de carga del modelo se muestran en la Figura 13: en la primera carga, el DFT 2 insertó 20 empleadores, el DFT 3 insertó 469 personas y el DFT 4 insertó 469 observaciones. Al ejecutar el paquete reestructurado sobre el modelo ya cargado, estos tres flujos no insertaron filas, porque los Lookups encontraron todos los registros.

**Figura 13**

*Ejecución de los flujos DFT 2, DFT 3 y DFT 4 del paquete A en la primera carga*



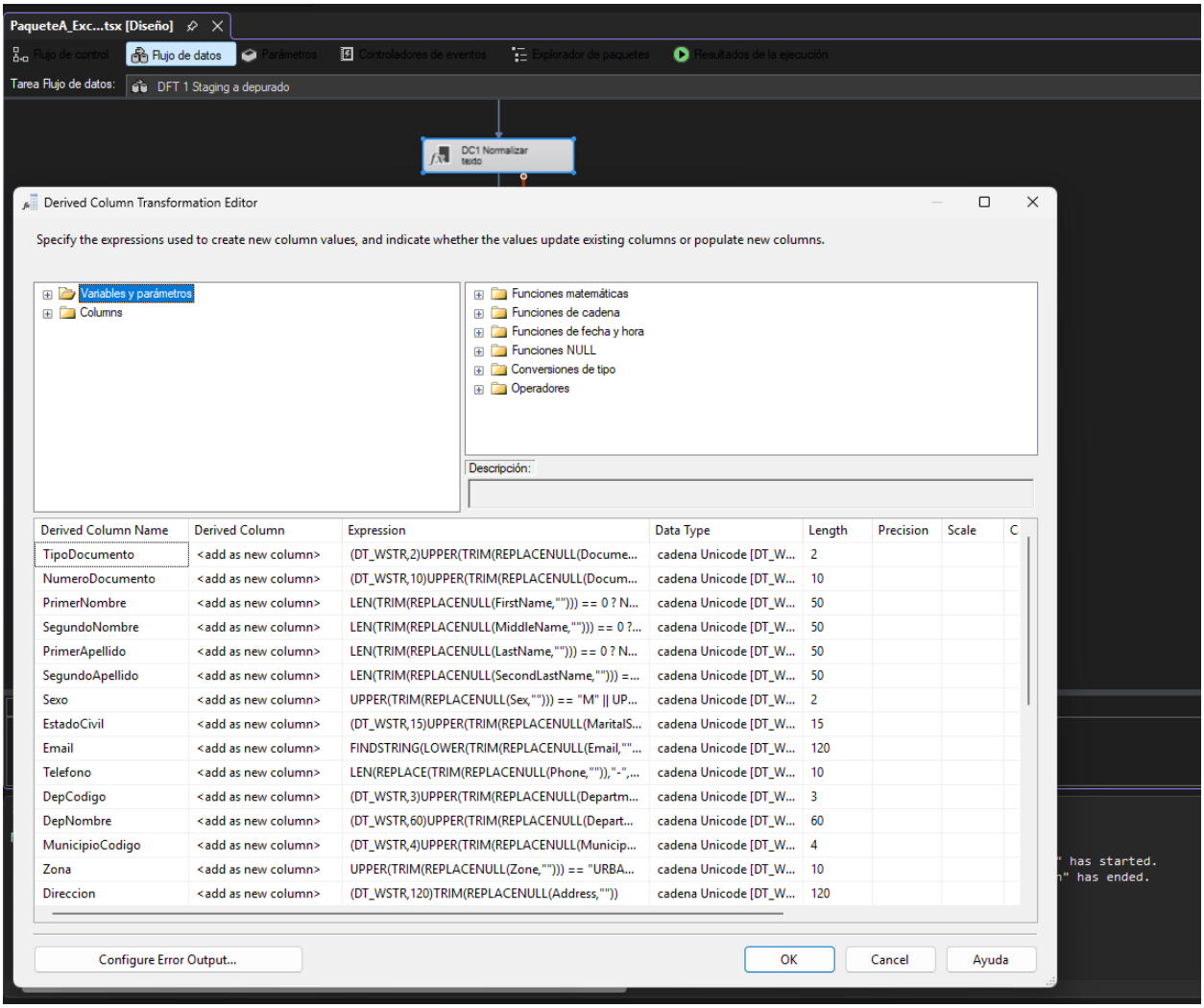



## Configuración de los componentes de transformación

Las Figuras 14 a 27 muestran la ventana de configuración de los componentes del DFT 1 que modifican los datos, abiertos con doble clic en Visual Studio. Allí se ven las expresiones, los tipos de destino, la configuración de errores y las condiciones con que cada componente transforma o separa las filas.

**Figura 14**

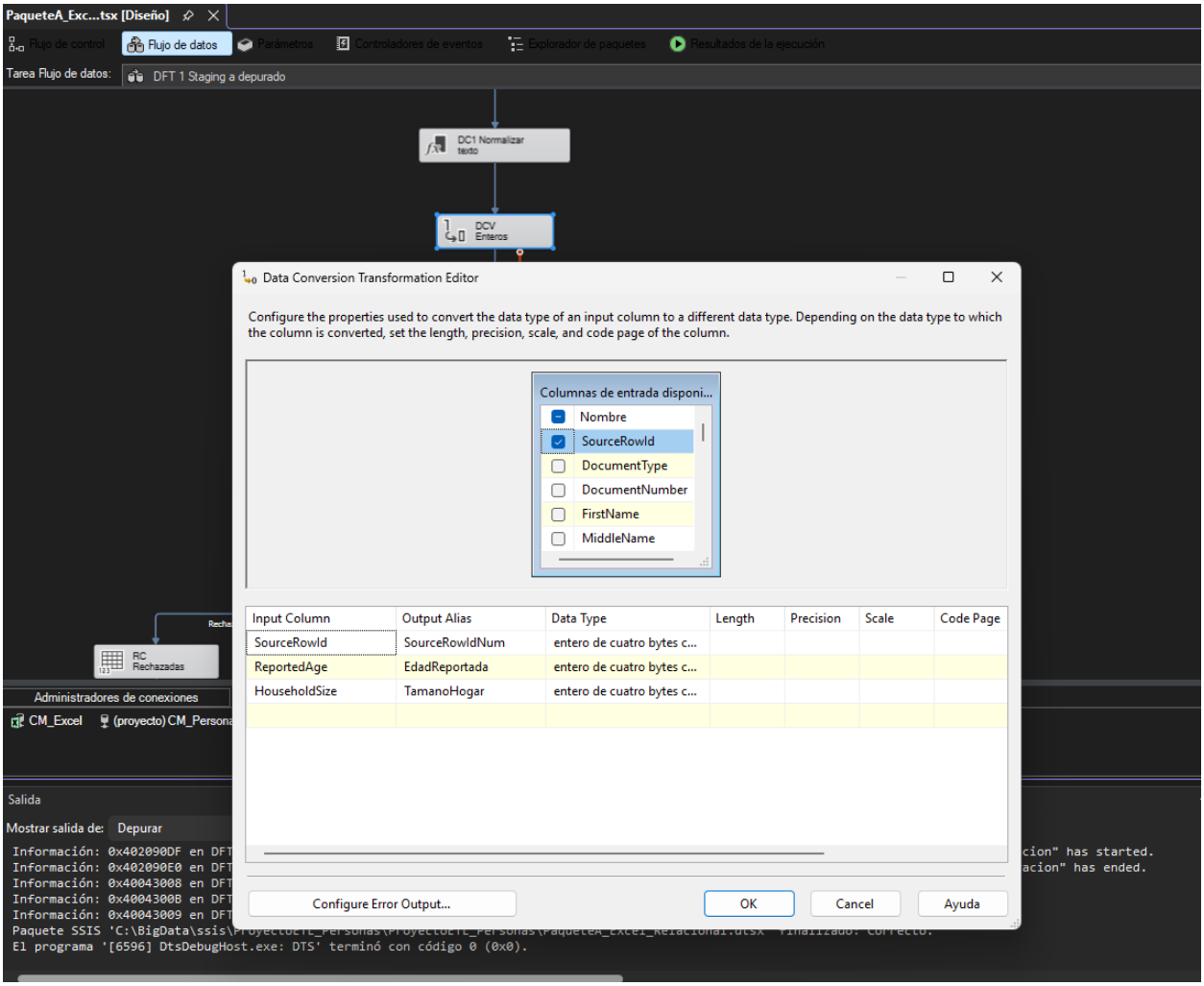
*DC1 Normalizar texto (Derived Column)*




*Nota.* Cada fila crea una columna limpia: tipo y número de documento en mayúsculas sin espacios, nombres en formato propio, sexo, estado civil, email en minúsculas, teléfono sin guiones, códigos geográficos, zona y dirección. Las expresiones completas están en ssis/expresiones_paquetes.txt.

**Figura 15**

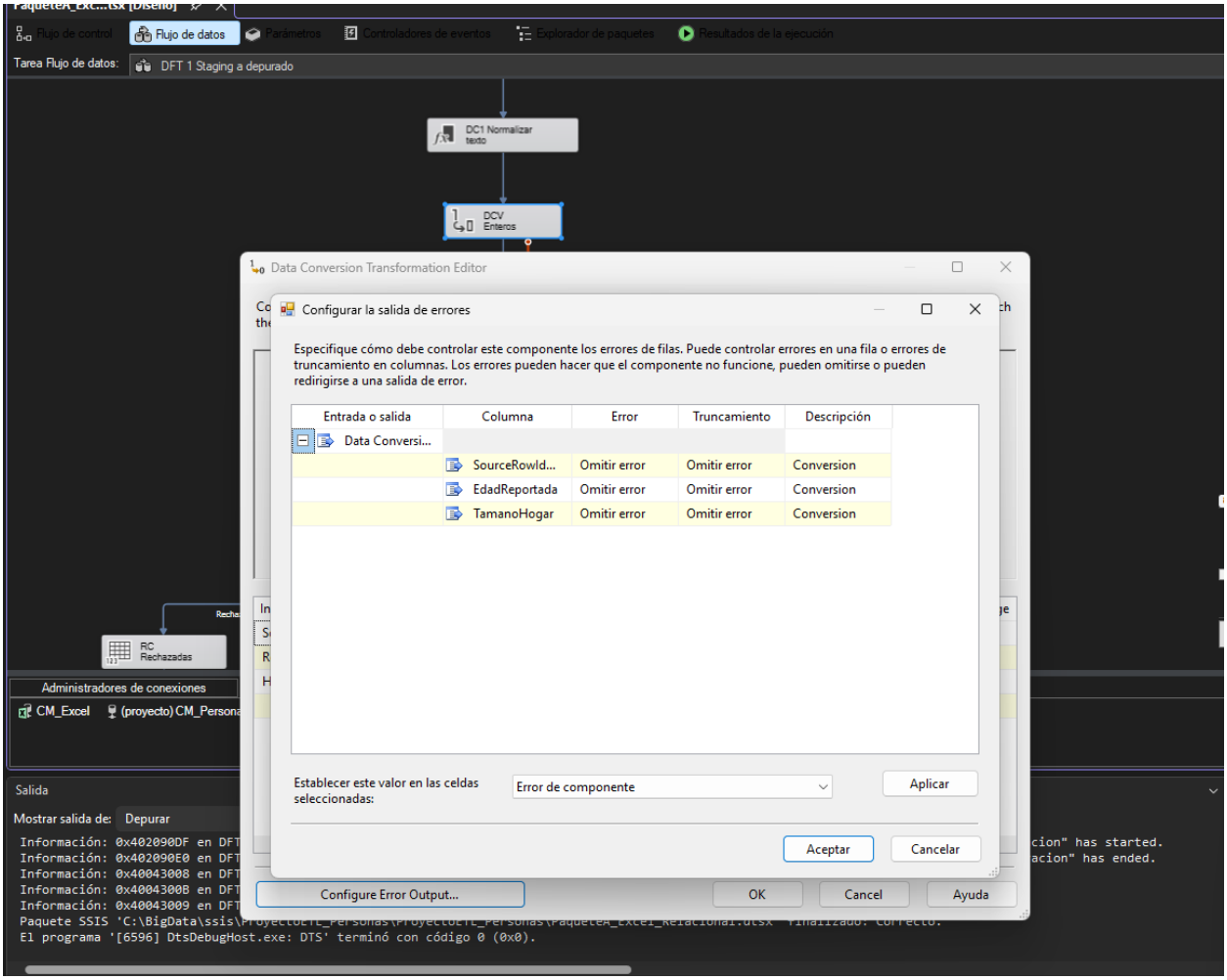
*DCV Enteros (Data Conversion)*




*Nota.* SourceRowId, ReportedAge y HouseholdSize se convierten de texto a entero de cuatro bytes (DT_I4) con los alias SourceRowIdNum, EdadReportada y TamanoHogar.

**Figura 16**

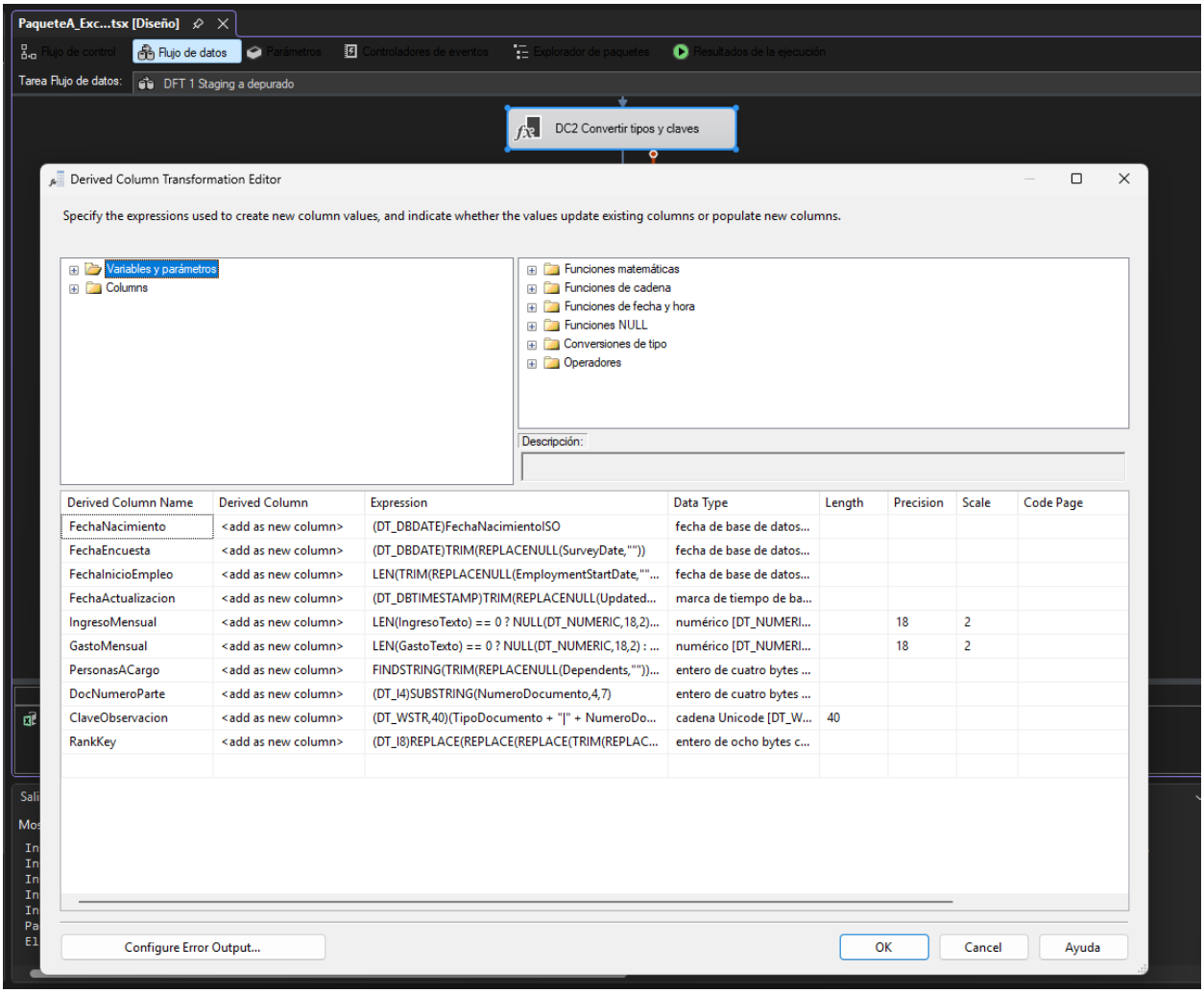
*Salida de error de DCV Enteros*




*Nota.* Error y Truncamiento en "Omitir error": si un valor no se puede convertir queda en NULL y las reglas de revisión lo detectan, en lugar de detener el paquete.

**Figura 17**

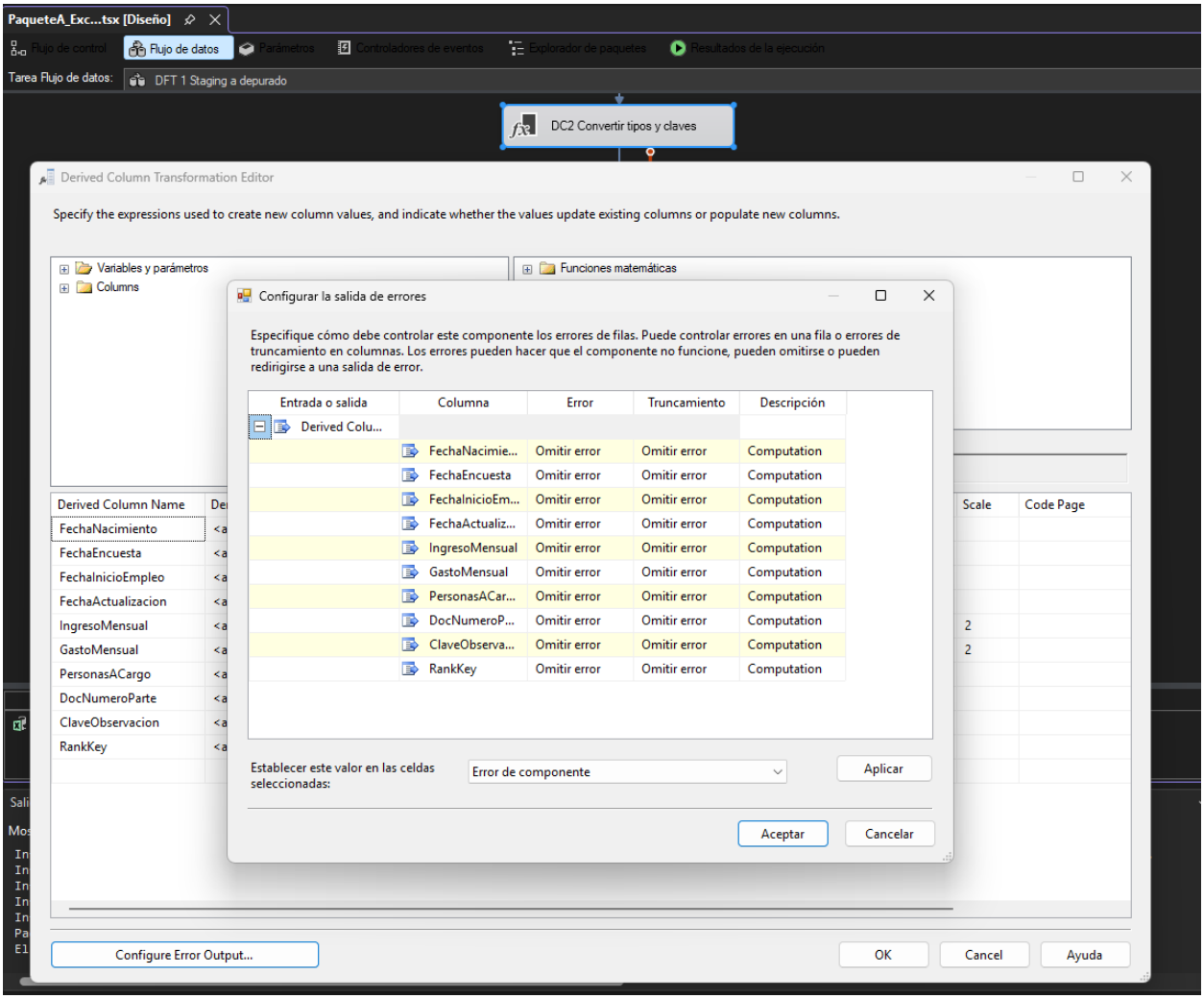
*DC2 Convertir tipos y claves (Derived Column)*




*Nota.* Convierte las fechas a DT_DBDATE y DT_DBTIMESTAMP, el ingreso y el gasto a DT_NUMERIC(18,2) y las personas a cargo a entero; además calcula ClaveObservacion (tipo, número y fecha de encuesta) y RankKey (DT_I8), que ordena las versiones de un duplicado.

**Figura 18**

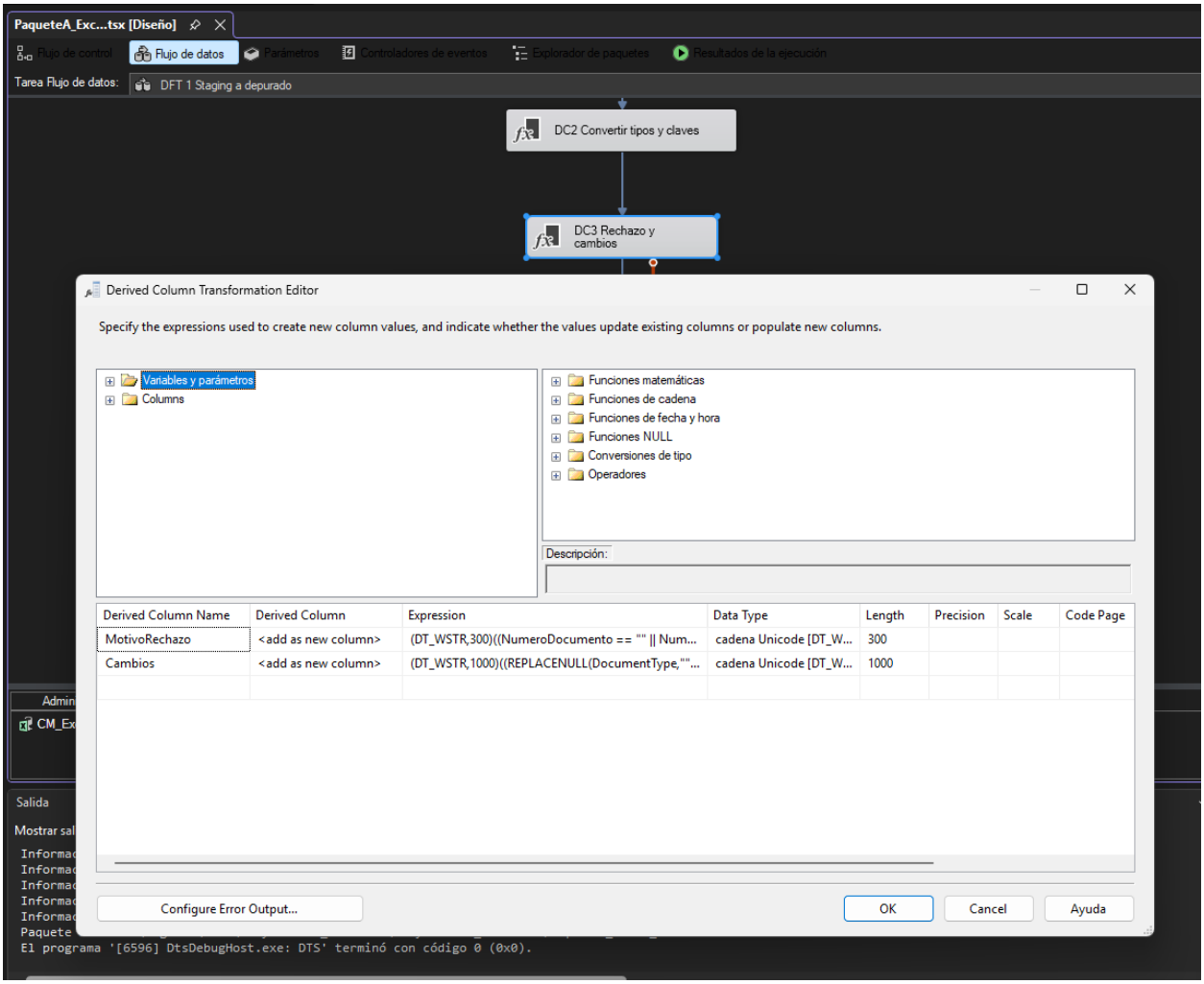
*Salida de error de DC2 Convertir tipos y claves*




*Nota.* Las diez columnas calculadas tienen "Omitir error" en Error y Truncamiento: una fecha imposible o un importe en letras quedan en NULL.

**Figura 19**

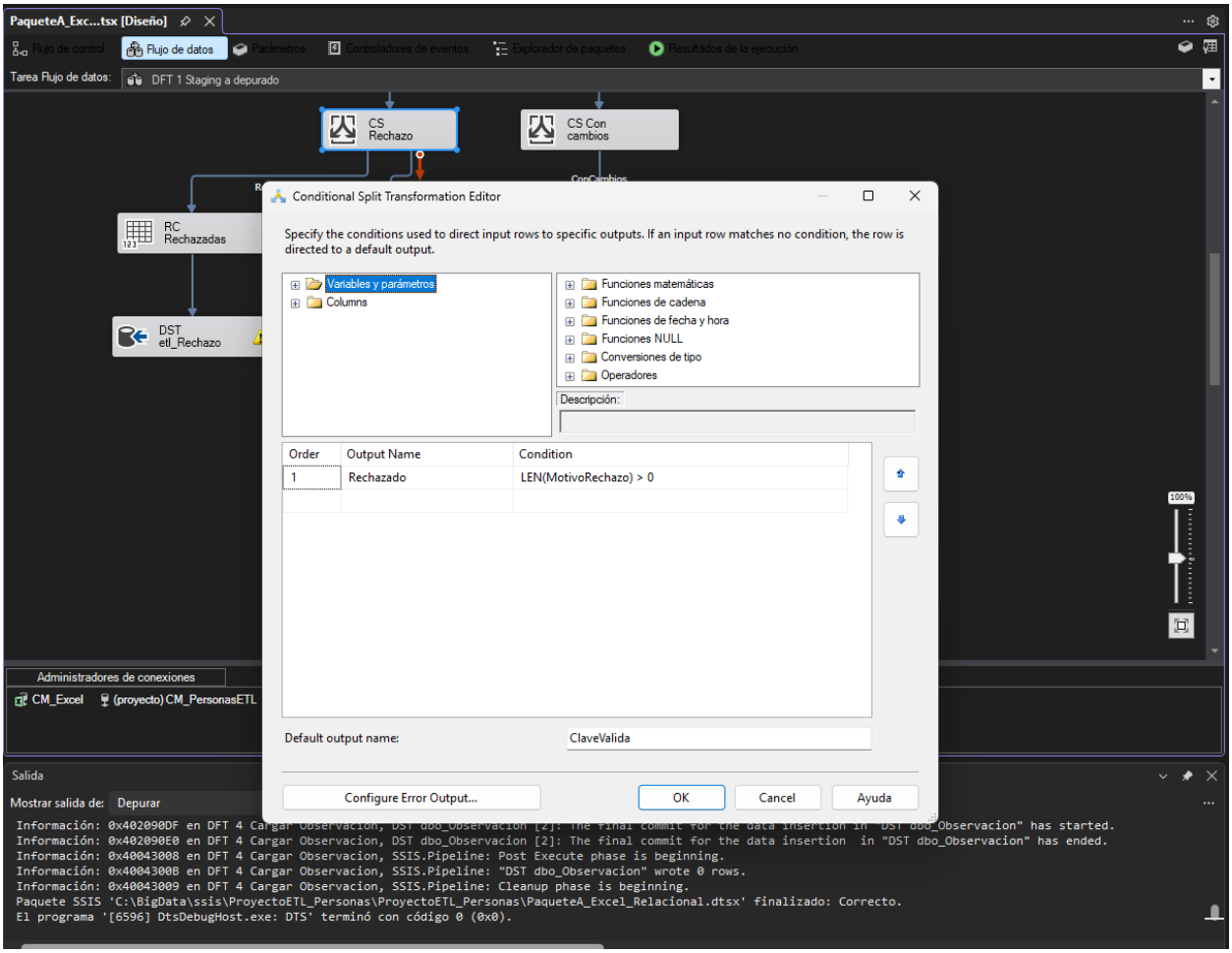
*DC3 Rechazo y cambios (Derived Column)*




*Nota.* MotivoRechazo acumula los motivos que impiden identificar el registro (documento, primer nombre o fecha de encuesta) y Cambios lista las correcciones aplicadas a la fila.

**Figura 20**

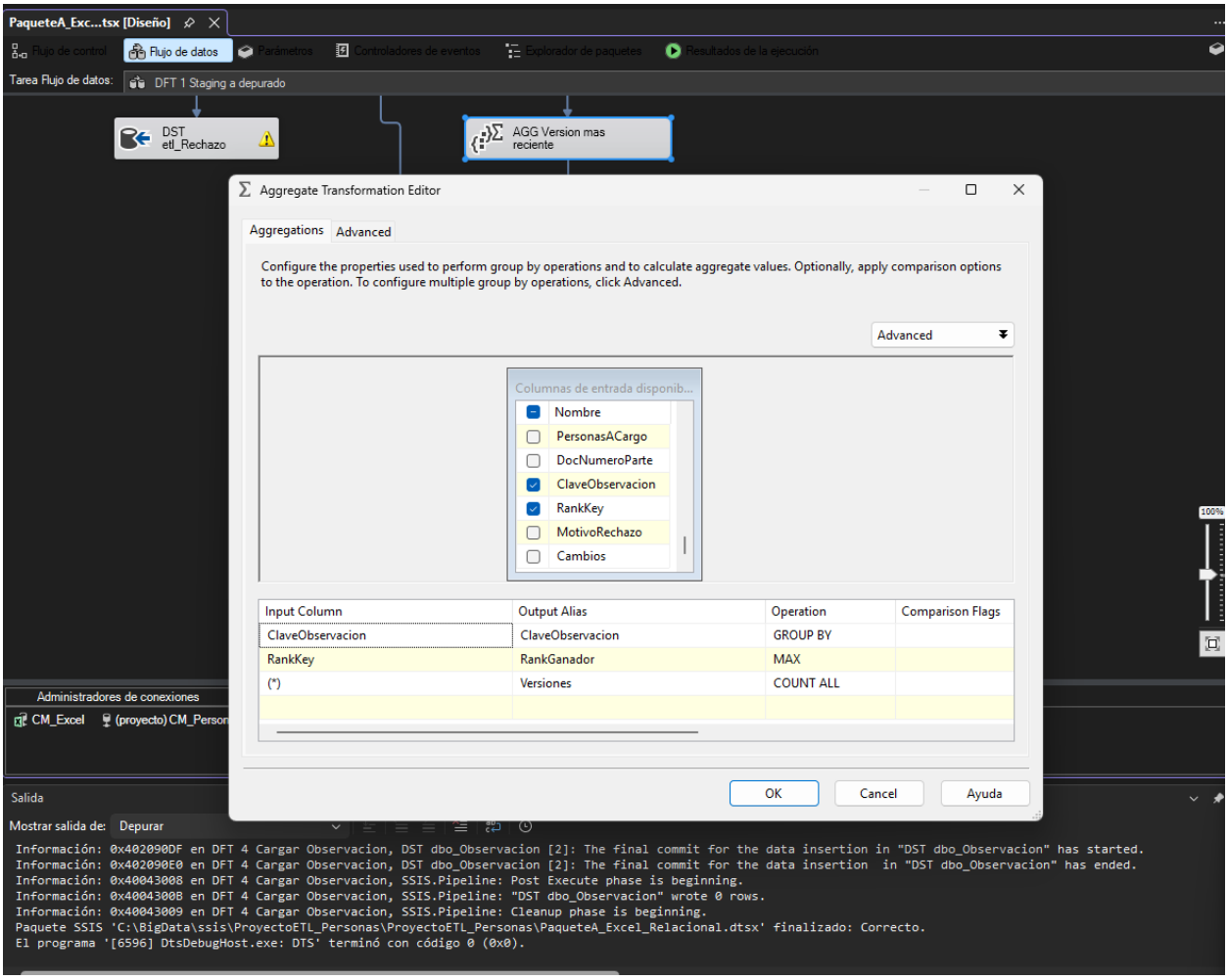
*CS Rechazo (Conditional Split)*




*Nota.* La condición LEN(MotivoRechazo) > 0 envía la fila a la salida Rechazado (etl.Rechazo); las demás siguen por la salida por defecto ClaveValida.

**Figura 21**

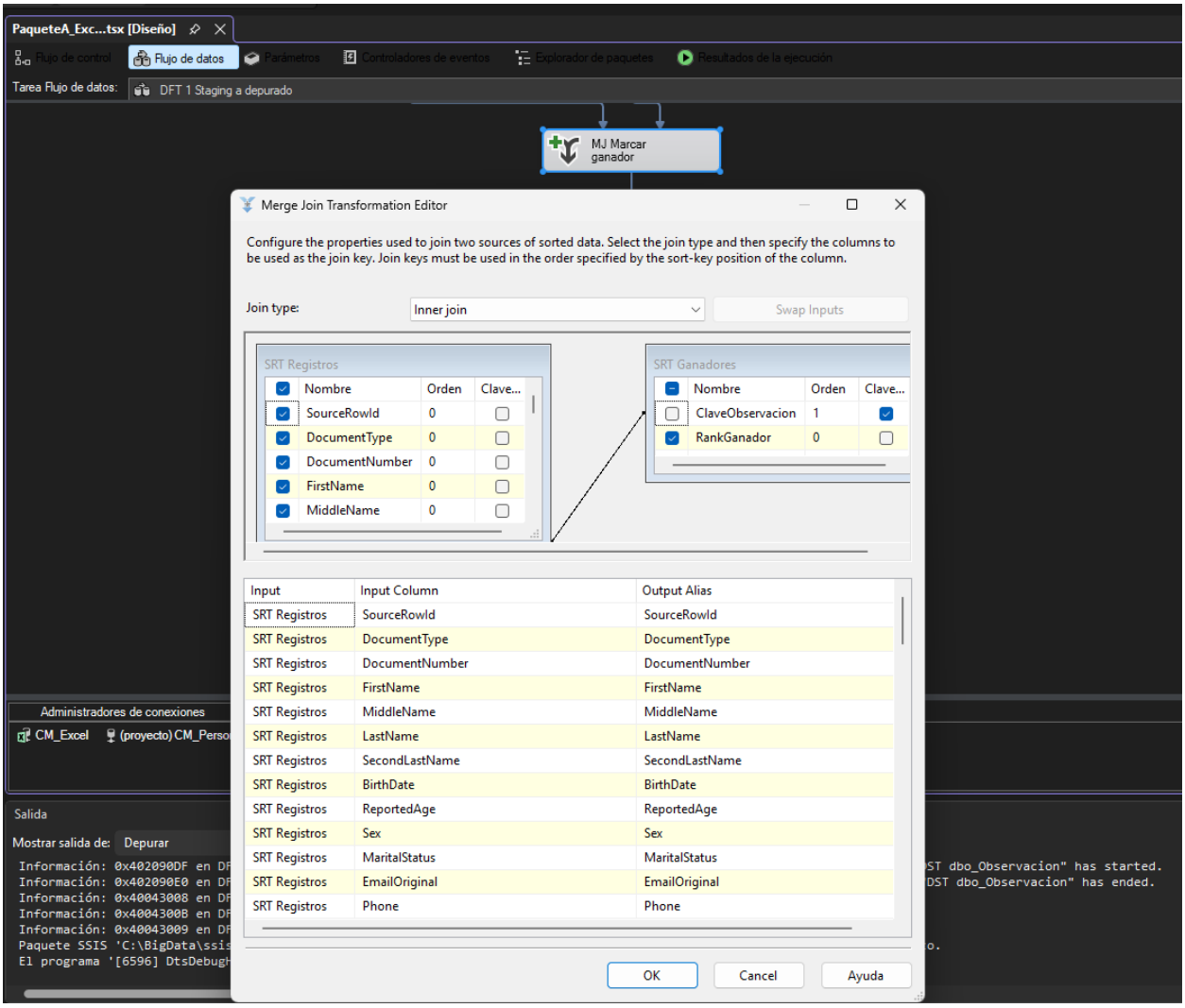
*AGG Version mas reciente (Aggregate)*




*Nota.* Agrupa por ClaveObservacion, calcula MAX(RankKey) como RankGanador y cuenta las versiones de cada observación.

**Figura 22**

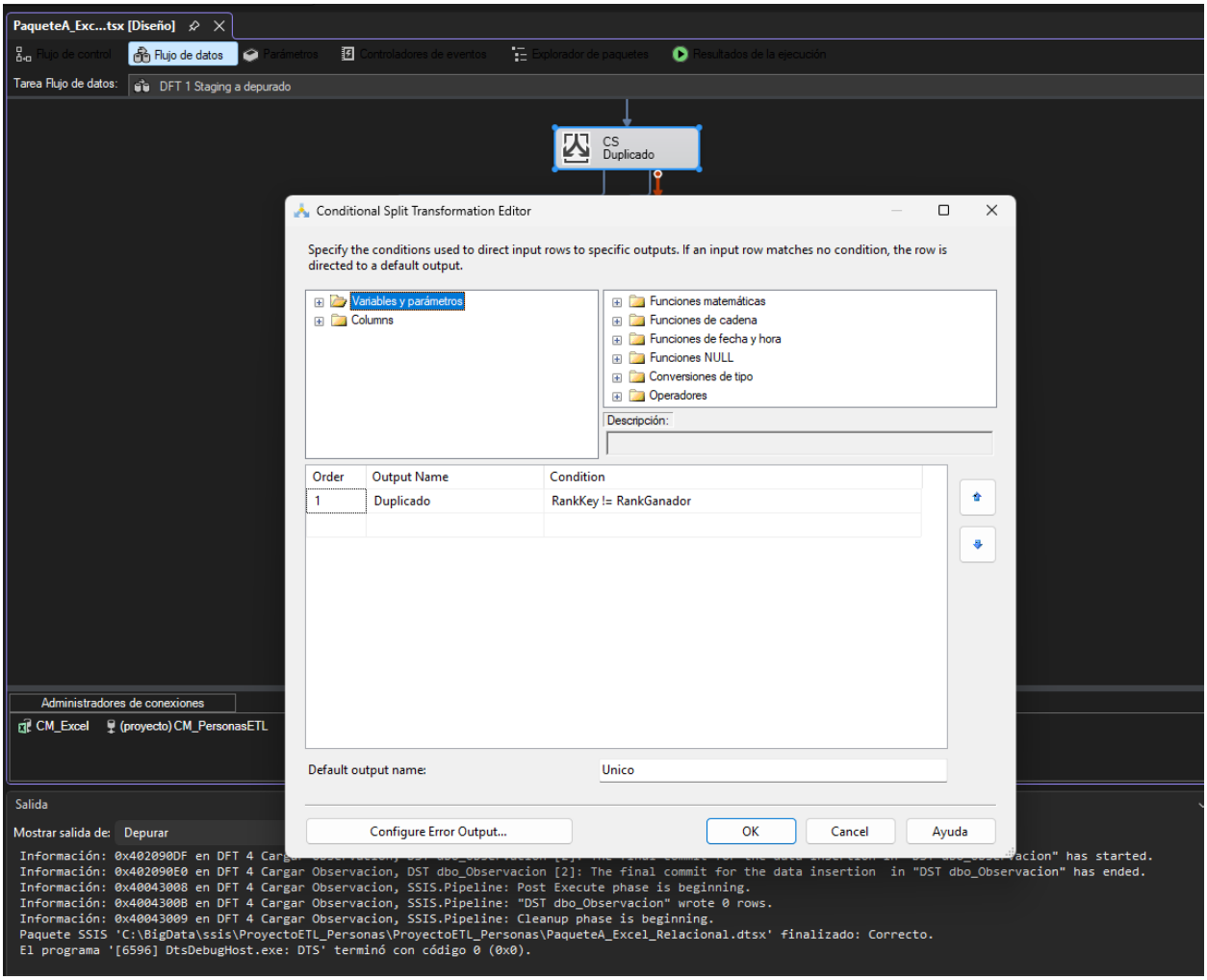
*MJ Marcar ganador (Merge Join)*




*Nota.* Combinación interna entre SRT Registros y SRT Ganadores por ClaveObservacion: cada fila recibe el RankGanador de su clave.

**Figura 23**

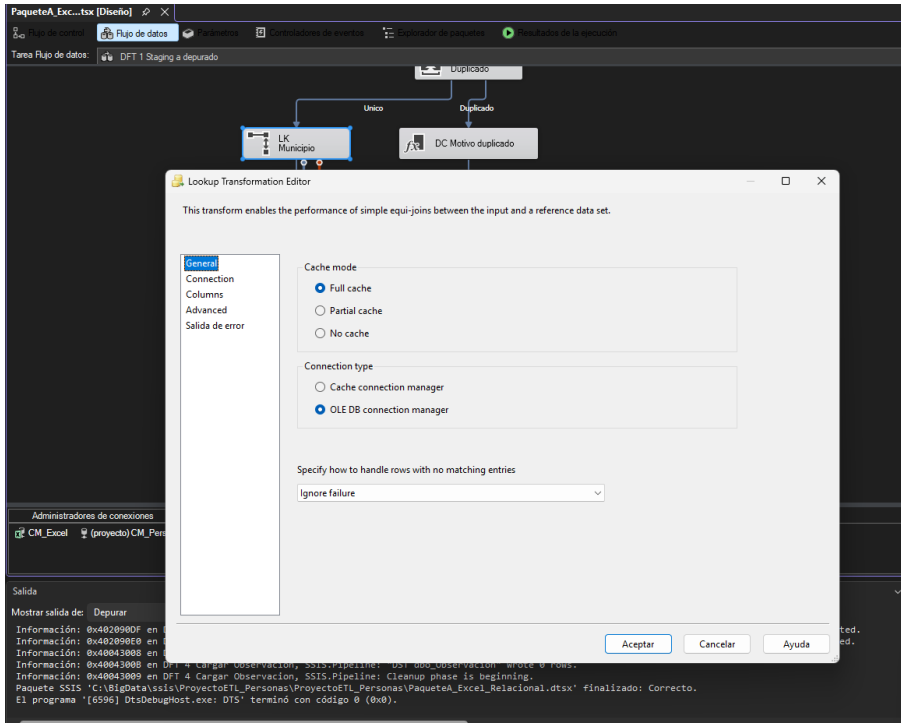
*CS Duplicado (Conditional Split)*

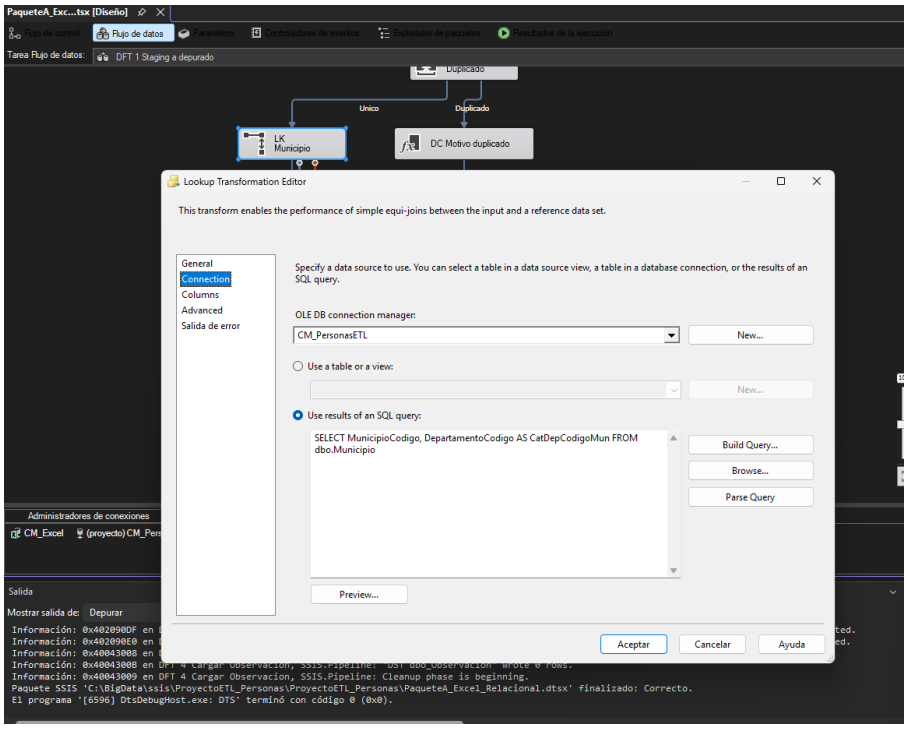



*Nota.* La condición RankKey != RankGanador envía a la salida Duplicado las versiones que no son la vigente; la versión ganadora sigue por la salida Unico.

**Figura 24**

*LK Municipio (Lookup): páginas General y Conexión*

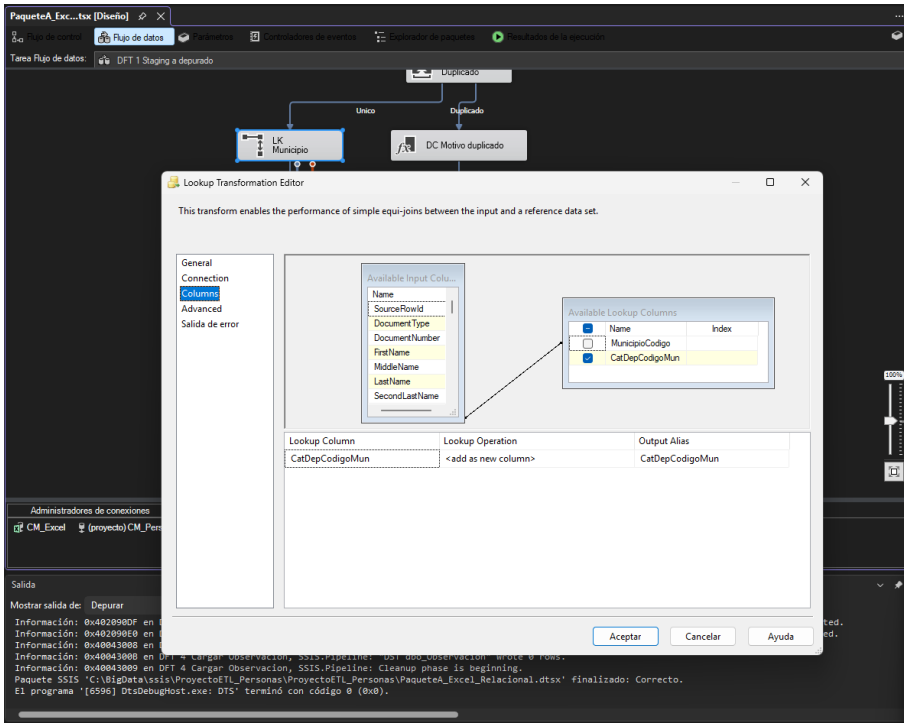


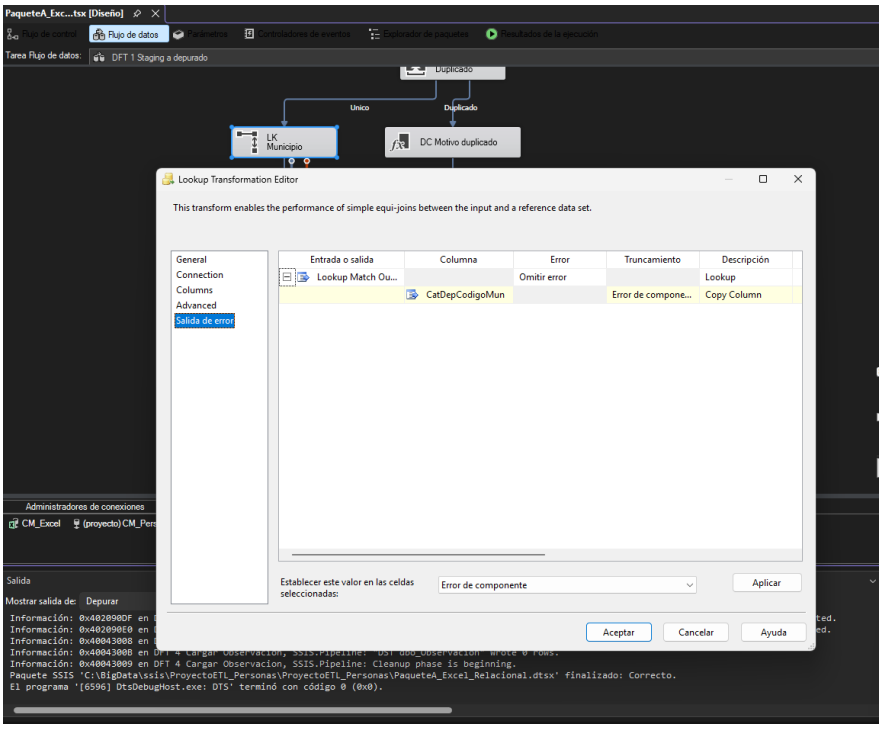




*Nota.* Caché completa sobre CM_PersonasETL y "Ignore failure" para las filas sin coincidencia. La consulta de referencia trae MunicipioCodigo y el DepartamentoCodigo de cada municipio.

**Figura 25**

*LK Municipio (Lookup): páginas Columnas y Salida de error*



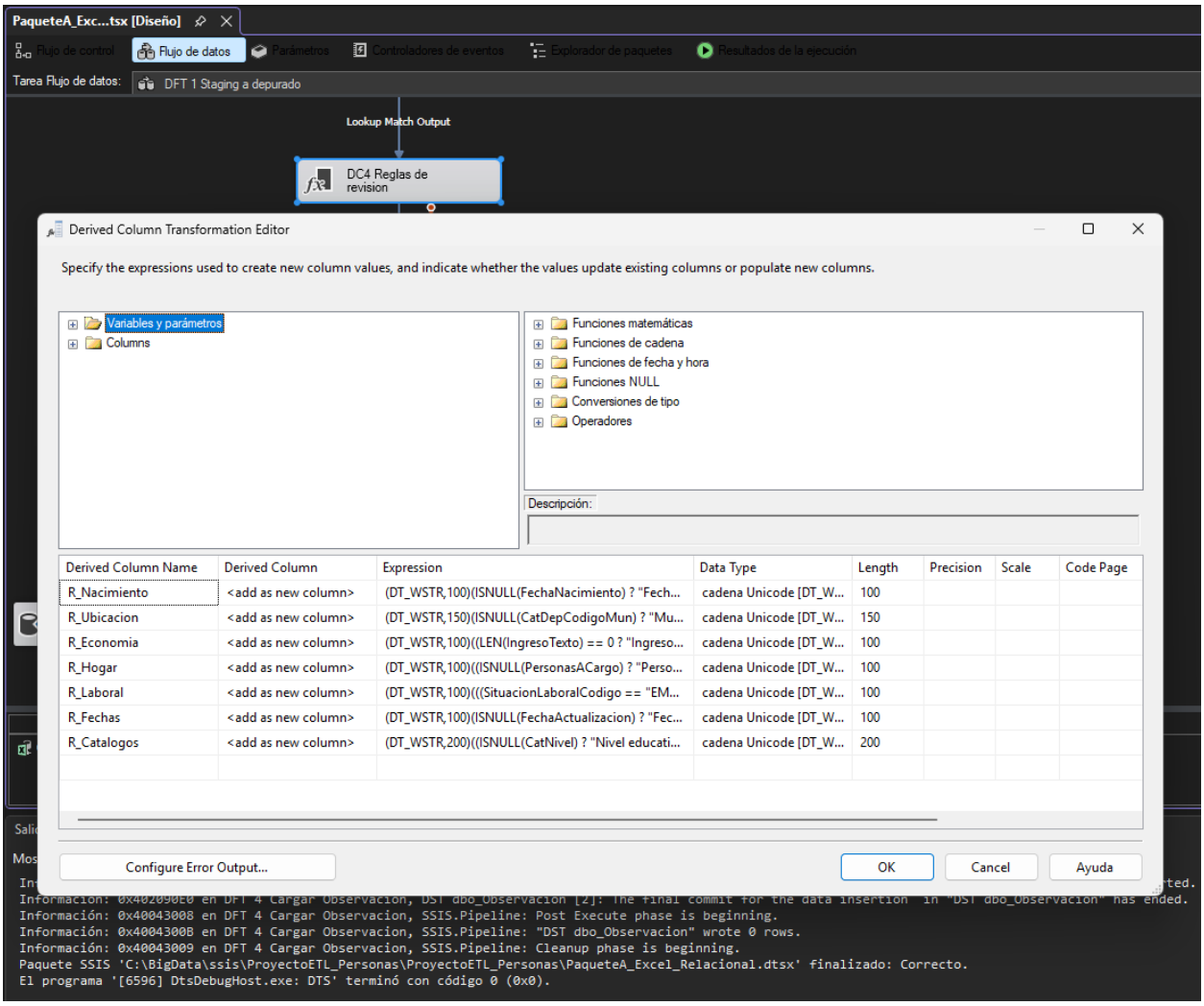




*Nota.* La fila se une por MunicipioCodigo y recibe CatDepCodigoMun. Si el municipio no existe (M999), la columna queda en NULL y DC4 envía la fila a revisión; si existe, DC4 compara el departamento del

catálogo con el informado.

**Figura 26**

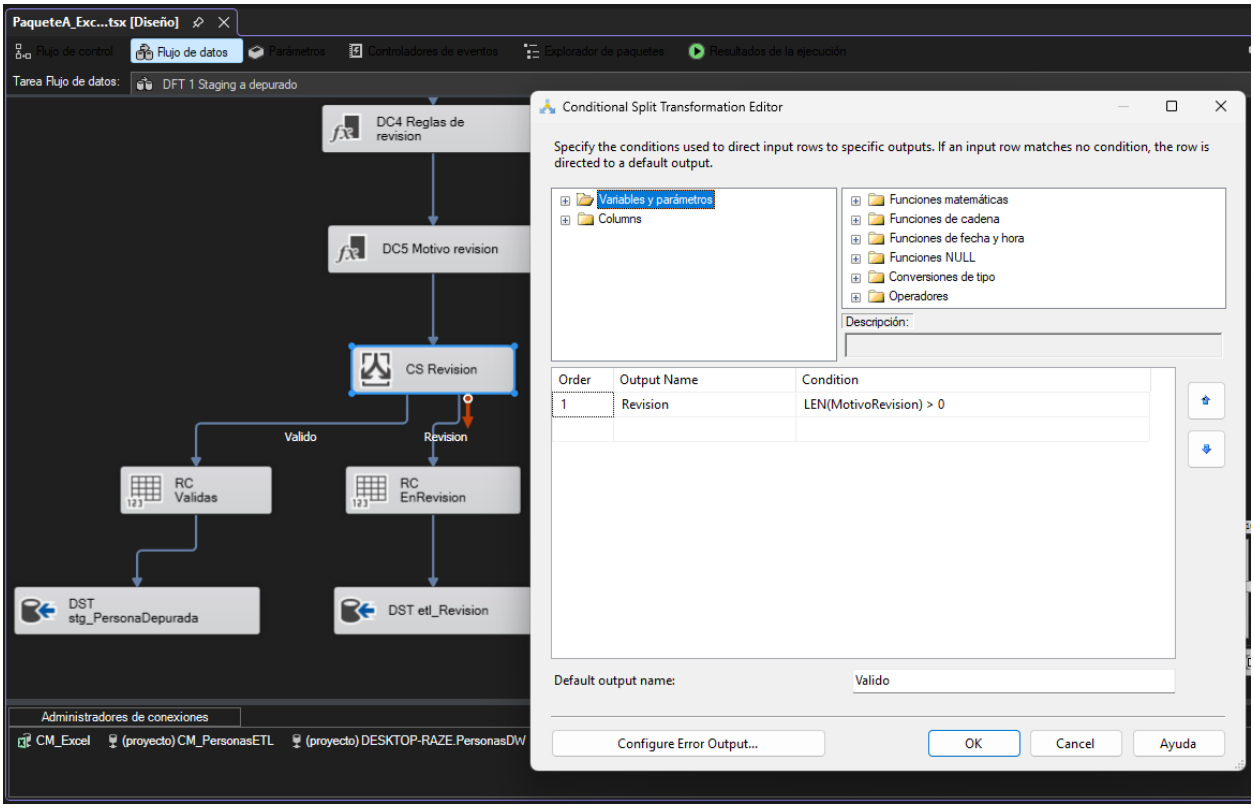
*DC4 Reglas de revision (Derived Column)*




*Nota.* Siete columnas de reglas (nacimiento, ubicación, economía, hogar, información laboral, fechas y catálogos); cada una devuelve el texto del motivo o vacío.

**Figura 27**

*CS Revision (Conditional Split)*




*Nota.* La condición LEN(MotivoRevision) > 0 envía la fila a etl.Revision; las que superan todas las reglas salen por Valido hacia stg.PersonaDepurada.

## Paquete SSIS B: modelo relacional hacia la bodega de datos

El origen del paquete B son exclusivamente las tablas depuradas del modelo relacional, nunca el Excel ni el staging. La granularidad de la tabla de hechos es una observación por persona y fecha de encuesta; por eso su clave primaria es (PersonaKey, TiempoKey). Las dimensiones permiten analizar el tiempo, la ubicación, la educación y la situación laboral, y DimPersona permite contar personas distintas. La Figura 28 presenta el esquema estrella y la Tabla 9 sus elementos.

**Figura 28**

*Bodega de datos PersonasDW en esquema estrella*

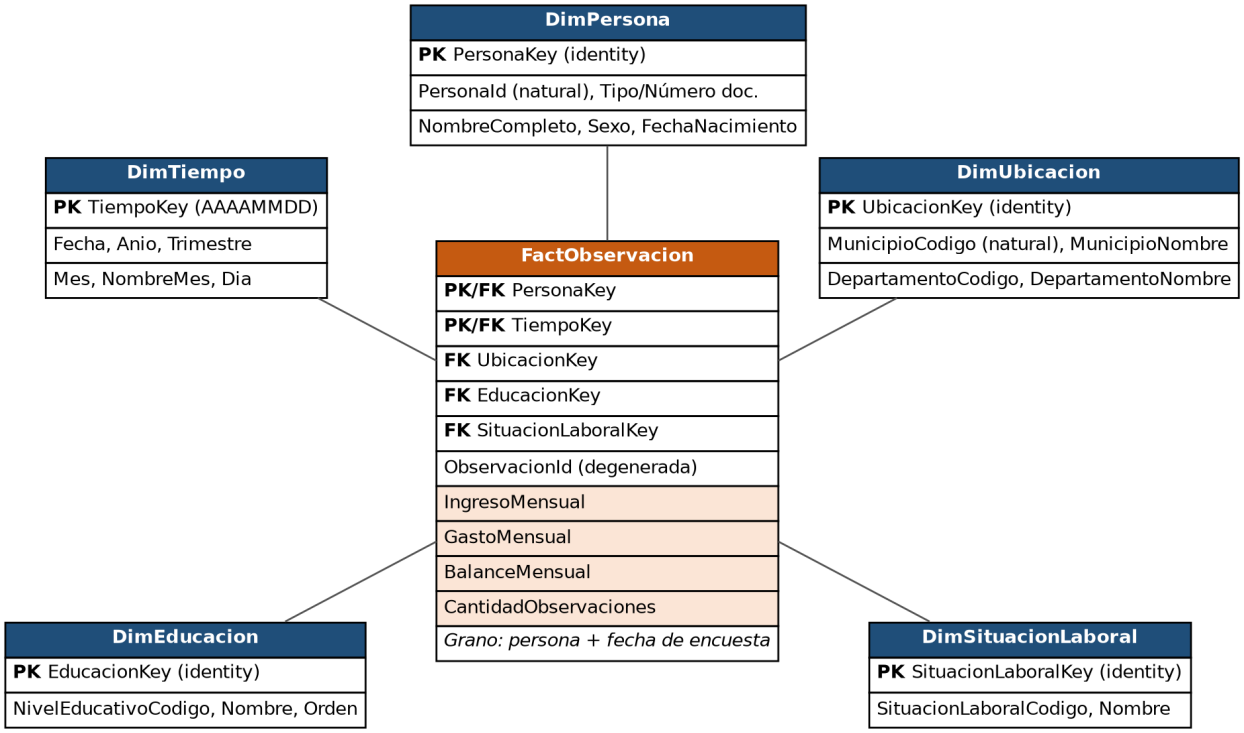



**Tabla 9**

*Dimensiones, hechos y claves de la bodega de datos*

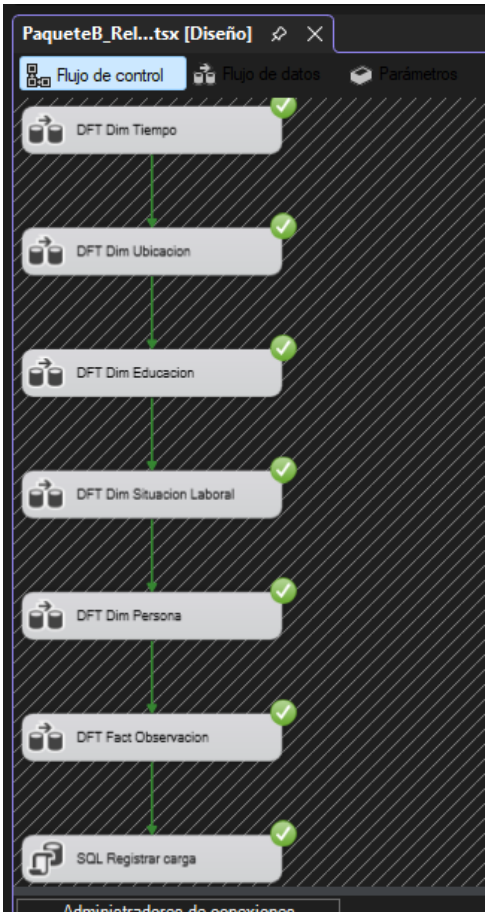
| Elemento | Contenido | Claves |
|---|---|---|
| DimTiempo | Fecha, año, trimestre, mes, nombre del mes y día | TiempoKey = AAAAMMDD |
| DimUbicacion | Municipio y departamento | UbicacionKey (identity); natural: MunicipioCodigo |
| DimEducacion | Nivel educativo y orden | EducacionKey (identity) |
| DimSituacionLaboral | Situación laboral | SituacionLaboralKey (identity) |
| DimPersona | Documento, nombre completo, sexo y nacimiento | PersonaKey (identity); natural: PersonaId |
| FactObservacion | IngresoMensual, GastoMensual, BalanceMensual y CantidadObservaciones | PK (PersonaKey, TiempoKey) |

Siguiendo a Kimball y Ross (2013), el paquete carga primero las cinco dimensiones y después los hechos (Figura 29). Cada dimensión lee su origen, busca el miembro en la

dimensión con un Lookup y solo inserta los que no existen. El flujo de hechos calcula TiempoKey, BalanceMensual y CantidadObservaciones, resuelve las claves sustitutas con cinco Lookups y verifica que el hecho no exista antes de insertarlo (Figura 30).

**Figura 29**

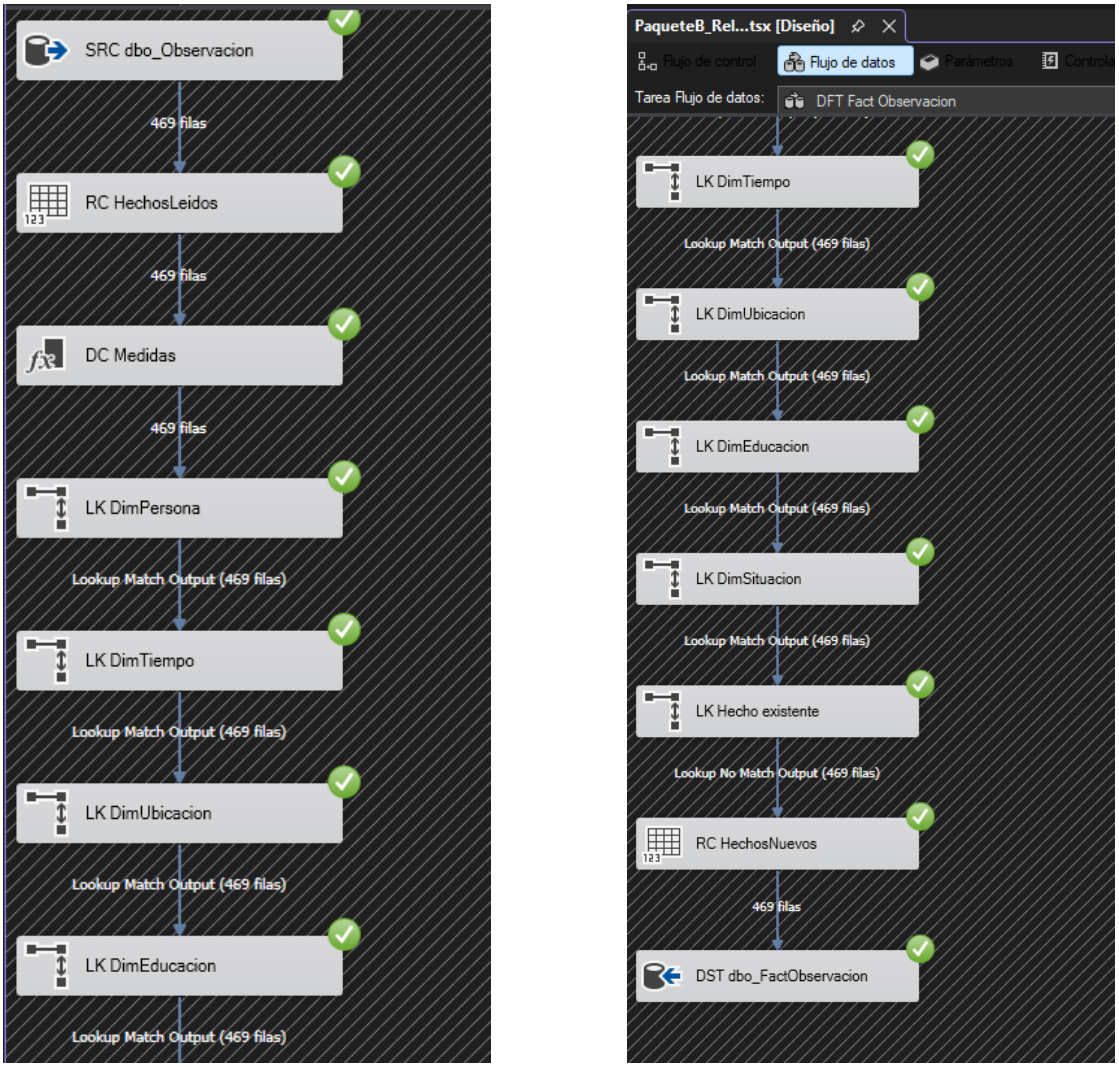
*Flujo de control del paquete B ejecutado*




*Nota.* Las cinco dimensiones se cargan antes que la tabla de hechos.

**Figura 30**

*Ejecución del flujo de hechos DFT Fact Observacion*

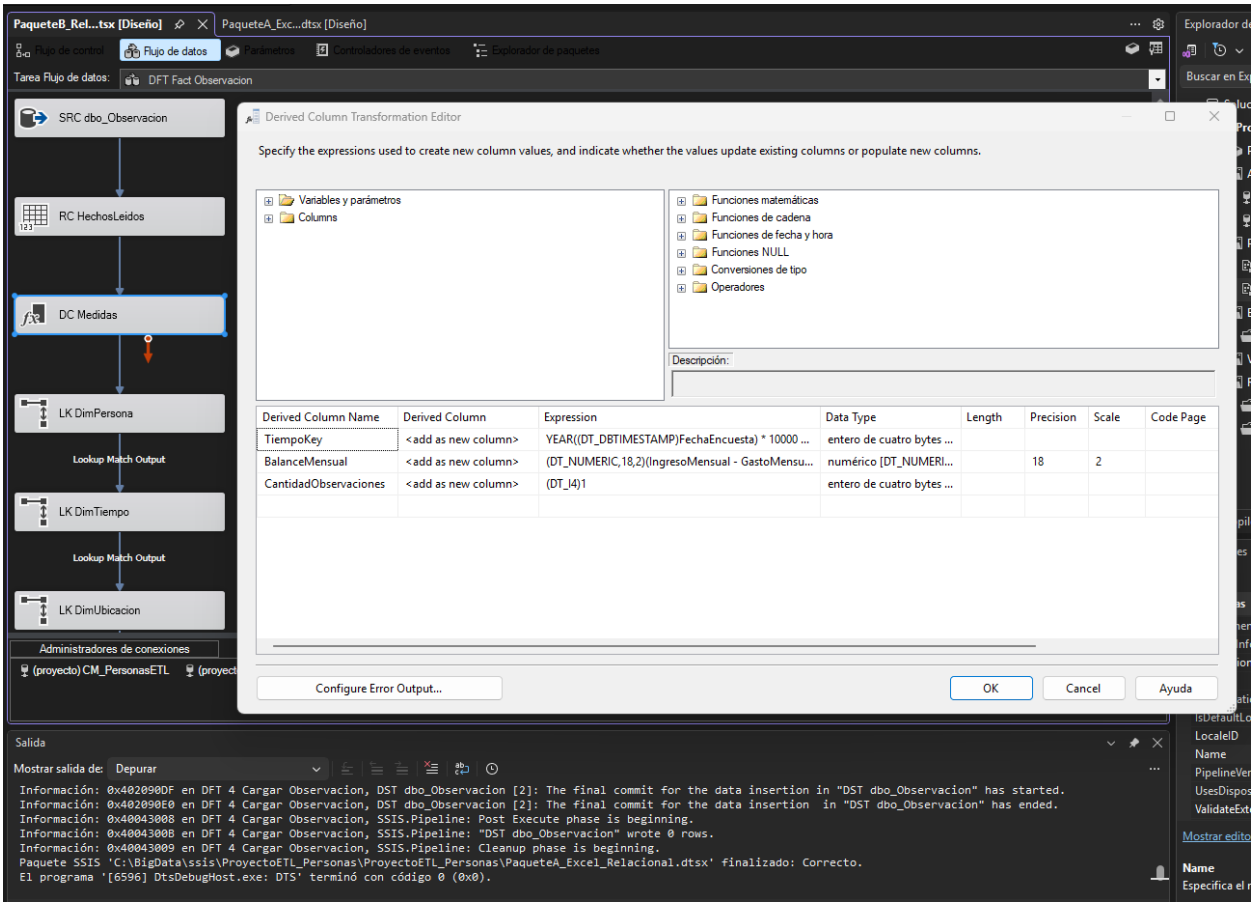



*Nota.* Las 469 observaciones encuentran sus claves sustitutas en las cinco dimensiones y se insertan como hechos nuevos.

La Figura 31 muestra la configuración de DC Medidas, el componente del flujo de hechos que calcula los datos nuevos.

**Figura 31**

*Configuración de DC Medidas (Derived Column) en el flujo DFT Fact Observacion*




*Nota.* TiempoKey = año × 10.000 + mes × 100 + día de la fecha de encuesta; BalanceMensual = ingreso menos gasto como DT_NUMERIC(18,2); CantidadObservaciones = 1, para poder sumar observaciones en las consultas.

## Resultados de la carga

La Tabla 10 resume la primera ejecución del paquete A. Cada fila del Excel terminó en un solo destino: 90 rechazadas, 50 duplicadas, 391 en revisión y 469 válidas, que suman 1.000.

**Tabla 10**

*Resultados de la carga del paquete A*

| Concepto | Registros |
|---|---|
| Leídos del Excel | 1000 |
| Rechazados | 90 |
| Descartados como duplicados | 50 |
| Enviados a revisión | 391 |
| Válidos cargados al modelo relacional | 469 |
| Personas / observaciones / empleadores en el modelo | 469 / 469 / 20 |
| Filas con al menos una corrección registrada | 509 |

## Motivos de rechazo y de revisión

La Tabla 11 presenta los motivos registrados. Un mismo registro puede tener varios motivos; la Figura 32 muestra la misma información consultada en SQL Server.

**Tabla 11**

*Motivos de rechazo y de revisión*

| Tipo | Motivo | Registros |
|---|---|---|
| Rechazo | Documento faltante | 68 |
| Rechazo | Fecha de encuesta inválida | 22 |
| Rechazo | Primer nombre faltante | 22 |
| Revisión | Información laboral inconsistente | 71 |
| Revisión | Edad inconsistente con nacimiento y encuesta | 46 |
| Revisión | Fecha de nacimiento inválida | 45 |
| Revisión | Municipio no existe en catálogo | 45 |
| Revisión | Ingreso no numérico | 44 |
| Revisión | Personas a cargo inválido | 44 |
| Revisión | Fecha de inicio de empleo incoherente | 30 |
| Revisión | Nacimiento posterior a la encuesta | 23 |
| Revisión | Departamento incoherente con municipio | 22 |
| Revisión | Nombre de departamento incoherente con código | 22 |
| Revisión | Ingreso negativo | 22 |
| Revisión | Gasto faltante | 22 |
| Revisión | Tamaño de hogar inválido | 22 |
| Revisión | Fecha de actualización anterior a la encuesta | 22 |

**Figura 32**

*Conteos de la ejecución y motivos de rechazo y revisión*

08_evidencia...\verqu (52)) | 06_consultas_...E\verqu (87))

```sql
/* ==================================================================
   08_evidencias_y_reejecucion.sql
   Ejecutar DESPUÉS de correr los paquetes SSIS (y otra vez tras la
   segunda ejecución). Produce las evidencias del informe:
     A. Conteos de aceptados, rechazados, revisión y duplicados
     B. Antes / después de registros de ejemplo
     C. Prueba de no duplicados tras la re-ejecución
   ------------------------------------------------------------------ */
```

100 % | No se encontraron problemas.

Resultados | Mensajes

| | Concepto | Registros |
|---|---|---|
| 1 | Leídas del Excel | 1000 |
| 2 | Rechazadas | 90 |
| 3 | Descartadas por duplicado | 50 |
| 4 | Enviadas a revisión | 391 |
| 5 | Válidas (depuradas) | 469 |
| 6 | Filas con correcciones | 509 |
| 7 | Personas en el modelo | 469 |
| 8 | Observaciones en el modelo | 469 |
| 9 | Empleadores en el modelo | 20 |

| | Tipo | Motivo | Registros |
|---|---|---|---|
| 1 | Rechazo | Documento faltante | 68 |
| 2 | Rechazo | Fecha de encuesta inválida | 22 |
| 3 | Rechazo | Primer nombre faltante | 22 |
| 4 | Revisión | Información laboral inconsistente | 71 |
| 5 | Revisión | Edad inconsistente con nacimiento y encuesta | 46 |
| 6 | Revisión | Municipio no existe en catálogo | 45 |
| 7 | Revisión | Fecha de nacimiento inválida | 45 |
| 8 | Revisión | Personas a cargo inválido | 44 |
| 9 | Revisión | Ingreso no numérico | 44 |
| 10 | Revisión | Fecha de inicio de empleo incoherente | 30 |
| 11 | Revisión | Nacimiento posterior a la encuesta | 23 |
| 12 | Revisión | Tamaño de hogar inválido | 22 |
| 13 | Revisión | Departamento incoherente con municipio | 22 |
| 14 | Revisión | Nombre de departamento incoherente con c... | 22 |
| 15 | Revisión | Ingreso negativo | 22 |
| 16 | Revisión | Gasto faltante | 22 |
| 17 | Revisión | Fecha de actualización anterior a la encuesta | 22 |

## Correcciones registradas

Las correcciones aplicadas a cada fila se guardan en etl.LogCambio (Tabla 12).

**Tabla 12**

*Correcciones registradas en etl.LogCambio*

| Corrección | Registros |
|---|---|
| Advertencia: personas a cargo > tamaño del hogar | 227 |
| Teléfono sin guiones | 45 |
| NivelEducativo a mayúsculas | 44 |
| EstadoCivil a mayúsculas | 44 |
| PrimerNombre estandarizado | 23 |
| PrimerApellido estandarizado | 23 |
| TipoDocumento a mayúsculas | 23 |
| FechaNacimiento dd/mm/aaaa a ISO | 23 |
| Discapacidad … a NO | 23 |
| Sexo inválido … a NULL | 22 |
| Email inválido a NULL | 22 |
| Teléfono inválido a NULL | 22 |
| Nombre de municipio tomado del catálogo | 22 |
| Estrato inválido … a NULL | 22 |
| Ingreso convertido de formato monetario | 22 |
| Gasto convertido de formato monetario | 22 |
| RegimenSalud … a NULL | 22 |
| Zona inválida … a NULL | 22 |
| Discapacidad … a SI | 21 |

*Nota.* La advertencia de personas a cargo mayores que el tamaño del hogar no bloquea el registro, porque una persona puede tener dependientes fuera del hogar.

## Evidencias antes y después del tratamiento

La Tabla 13 compara valores originales y tratados. Los casos de corrección, rechazo, revisión y duplicado se resumen en la Tabla 14, y la Figura 33 muestra la consulta correspondiente en SQL Server.

**Tabla 13**

*Valores antes y después del tratamiento*

| Fila | Campo | Antes (Excel) | Después (modelo) |
|---|---|---|---|
| 4 | DocumentType | cc | CC |
| 4 | Phone | 000-0000004 | 0000000004 |
| 1* | FirstName / LastName | andrés / GARCÍA | Andrés / García |
| 18 | MonthlyIncome | $ 14.700.000,00 | 14.700.000,00 |
| 22 | MonthlyExpenses | 8900000,50 | 8.900.000,50 |
| 22 | Disability | no | NO |
| 7* | BirthDate | 19/07/1967 | 1967-07-19 |
| 9* | Sex | Z | NULL |
| 10* | Email | persona10example.com | NULL |
| 16* | SocioeconomicStratum | ALTO | NULL |

*Nota.* * La corrección se aplicó y quedó registrada, aunque esa fila terminó en revisión por otra regla.

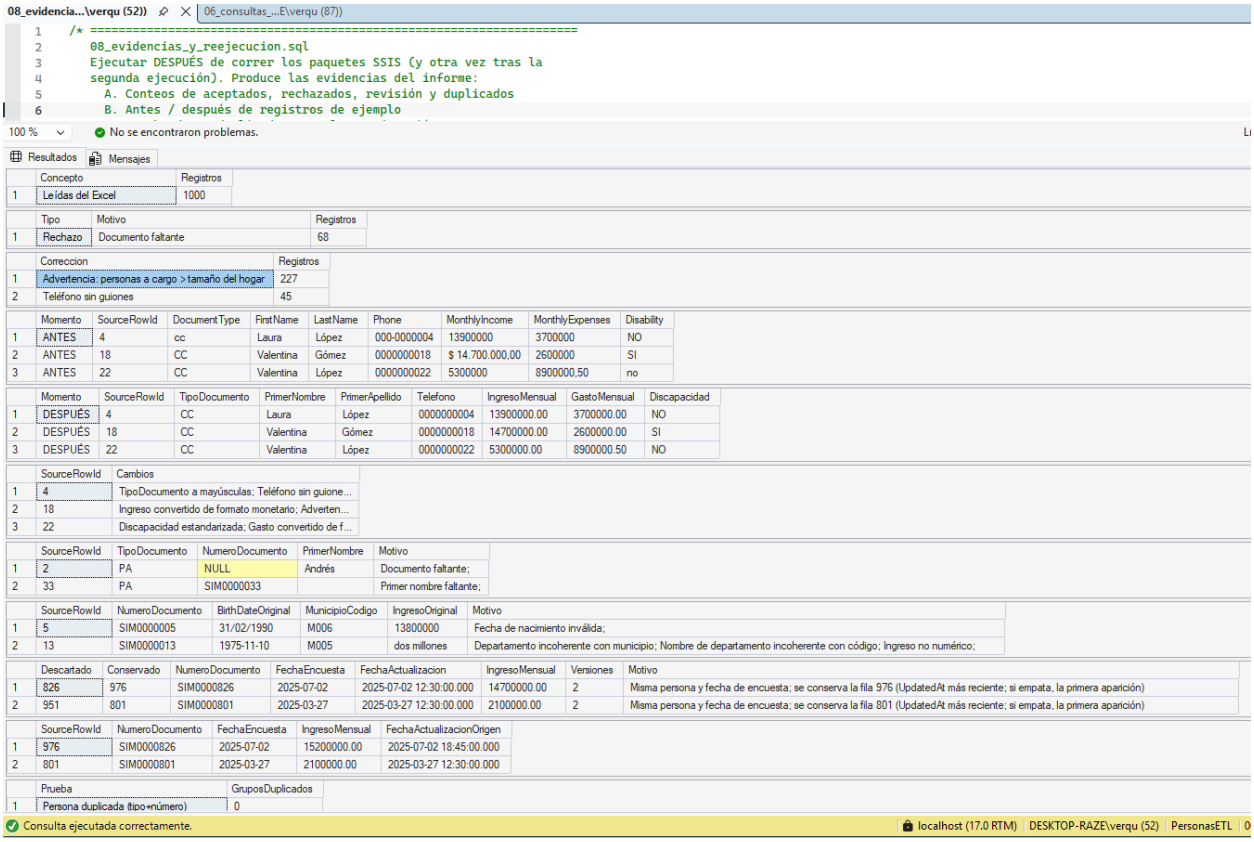
**Tabla 14**

*Ejemplos de corrección, rechazo, revisión y duplicado*

| Caso | Fila(s) | Detalle | Destino |
|---|---|---|---|
| Corrección | 4 | "cc" pasa a "CC" y "000-0000004" a "0000000004" | stg.PersonaDepurada |
| Rechazo | 2 | DocumentNumber vacío: documento faltante | etl.Rechazo |
| Rechazo | 33 | FirstName vacío: primer nombre faltante | etl.Rechazo |
| Revisión | 5 | BirthDate 31/02/1990: fecha de nacimiento inválida | etl.Revision |
| Revisión | 13 | Departamento incoherente con el municipio e ingreso "dos millones" | etl.Revision |
| Duplicado | 826 / 976 | Mismo documento y fecha; se conserva 976 (UpdatedAt 18:45, ingreso 15.200.000) | etl.Duplicado (826) |
| Duplicado | 801 / 951 | Copia idéntica; se conserva la primera aparición, 801 | etl.Duplicado (951) |

**Figura 33**

*Evidencias antes y después, rechazos, revisiones y duplicados en SQL Server*



```
/* ==================================================================
   08_evidencias_y_reejecucion.sql
   Ejecutar DESPUÉS de correr los paquetes SSIS (y otra vez tras la
   segunda ejecución). Produce las evidencias del informe:
     A. Conteos de aceptados, rechazados, revisión y duplicados
     B. Antes / después de registros de ejemplo
```

No se encontraron problemas.

Resultados | Mensajes

| | Concepto | Registros |
|---|---|---|
| 1 | Leídas del Excel | 1000 |

| | Tipo | Motivo | Registros |
|---|---|---|---|
| 1 | Rechazo | Documento faltante | 68 |

| | Correccion | Registros |
|---|---|---|
| 1 | Advertencia: personas a cargo > tamaño del hogar | 227 |
| 2 | Teléfono sin guiones | 45 |

| | Momento | SourceRowId | DocumentType | FirstName | LastName | Phone | MonthlyIncome | MonthlyExpenses | Disability |
|---|---|---|---|---|---|---|---|---|---|
| 1 | ANTES | 4 | cc | Laura | López | 000-0000004 | 13900000 | 3700000 | NO |
| 2 | ANTES | 18 | CC | Valentina | Gómez | 0000000018 | $ 14.700.000,00 | 2600000 | SI |
| 3 | ANTES | 22 | CC | Valentina | López | 0000000022 | 5300000 | 8900000,50 | no |

| | Momento | SourceRowId | TipoDocumento | PrimerNombre | PrimerApellido | Telefono | IngresoMensual | GastoMensual | Discapacidad |
|---|---|---|---|---|---|---|---|---|---|
| 1 | DESPUÉS | 4 | CC | Laura | López | 0000000004 | 13900000.00 | 3700000.00 | NO |
| 2 | DESPUÉS | 18 | CC | Valentina | Gómez | 0000000018 | 14700000.00 | 2600000.00 | SI |
| 3 | DESPUÉS | 22 | CC | Valentina | López | 0000000022 | 5300000.00 | 8900000.50 | NO |

| | SourceRowId | Cambios |
|---|---|---|
| 1 | 4 | TipoDocumento a mayúsculas; Teléfono sin guione... |
| 2 | 18 | Ingreso convertido de formato monetario; Adverten... |
| 3 | 22 | Discapacidad estandarizada; Gasto convertido de f... |

| | SourceRowId | TipoDocumento | NumeroDocumento | PrimerNombre | Motivo |
|---|---|---|---|---|---|
| 1 | 2 | PA | NULL | Andrés | Documento faltante; |
| 2 | 33 | PA | SIM0000033 | | Primer nombre faltante; |

| | SourceRowId | NumeroDocumento | BirthDateOriginal | MunicipioCodigo | IngresoOriginal | Motivo |
|---|---|---|---|---|---|---|
| 1 | 5 | SIM0000005 | 31/02/1990 | M006 | 13800000 | Fecha de nacimiento inválida; |
| 2 | 13 | SIM0000013 | 1975-11-10 | M005 | dos millones | Departamento incoherente con municipio; Nombre de departamento incoherente con código; Ingreso no numérico; |

| | Descartado | Conservado | NumeroDocumento | FechaEncuesta | FechaActualizacion | IngresoMensual | Versiones | Motivo |
|---|---|---|---|---|---|---|---|---|
| 1 | 826 | 976 | SIM0000826 | 2025-07-02 | 2025-07-02 12:30:00.000 | 14700000.00 | 2 | Misma persona y fecha de encuesta; se conserva la fila 976 (UpdatedAt más reciente; si empata, la primera aparición) |
| 2 | 951 | 801 | SIM0000801 | 2025-03-27 | 2025-03-27 12:30:00.000 | 2100000.00 | 2 | Misma persona y fecha de encuesta; se conserva la fila 801 (UpdatedAt más reciente; si empata, la primera aparición) |

| | SourceRowId | NumeroDocumento | FechaEncuesta | IngresoMensual | FechaActualizacionOrigen |
|---|---|---|---|---|---|
| 1 | 976 | SIM0000826 | 2025-07-02 | 15200000.00 | 2025-07-02 18:45:00.000 |
| 2 | 801 | SIM0000801 | 2025-03-27 | 2100000.00 | 2025-03-27 12:30:00.000 |

| | Prueba | GruposDuplicados |
|---|---|---|
| 1 | Persona duplicada (tipo+número) | 0 |

Consulta ejecutada correctamente. | localhost (17.0 RTM) | DESKTOP-RAZE\verqu (52) | PersonasETL

## Consultas analíticas y verificación

Las cuatro consultas se ejecutaron sobre la bodega de datos con el script *06_consultas_dw_y_validacion.sql*. Sus resultados aparecen en las Tablas 15 a 18 y en la Figura 34.

**Tabla 15**

*Personas distintas por municipio*

| Municipio | Personas distintas |
|---|---|
| CHÍA | 49 |
| CAJICÁ | 45 |
| ZIPAQUIRÁ | 43 |
| SOACHA | 53 |
| BOGOTÁ D.C. | 51 |
| MEDELLÍN | 47 |
| ENVIGADO | 49 |
| TUNJA | 46 |
| DUITAMA | 46 |
| CALI | 40 |

**Tabla 16**

*Ingreso mensual promedio por nivel educativo*

| Nivel educativo | Ingreso promedio (COP) |
|---|---|
| NINGUNO | 8.320.689,66 |
| PRIMARIA | 7.449.253,73 |
| SECUNDARIA | 8.426.086,96 |
| TECNICO | 7.383.333,33 |
| TECNOLOGO | 7.667.213,11 |
| UNIVERSITARIO | 7.593.650,79 |
| POSGRADO | 7.196.428,57 |

**Tabla 17**

*Balance mensual promedio por situación laboral*

| Situación laboral | Balance promedio (COP) |
|---|---|
| DESEMPLEADO | 2.283.999,98 |
| EMPLEADO | 2.068.749,96 |
| INDEPENDIENTE | 1.551.724,12 |
| ESTUDIANTE | 1.094.186,02 |
| PENSIONADO | 880.999,98 |

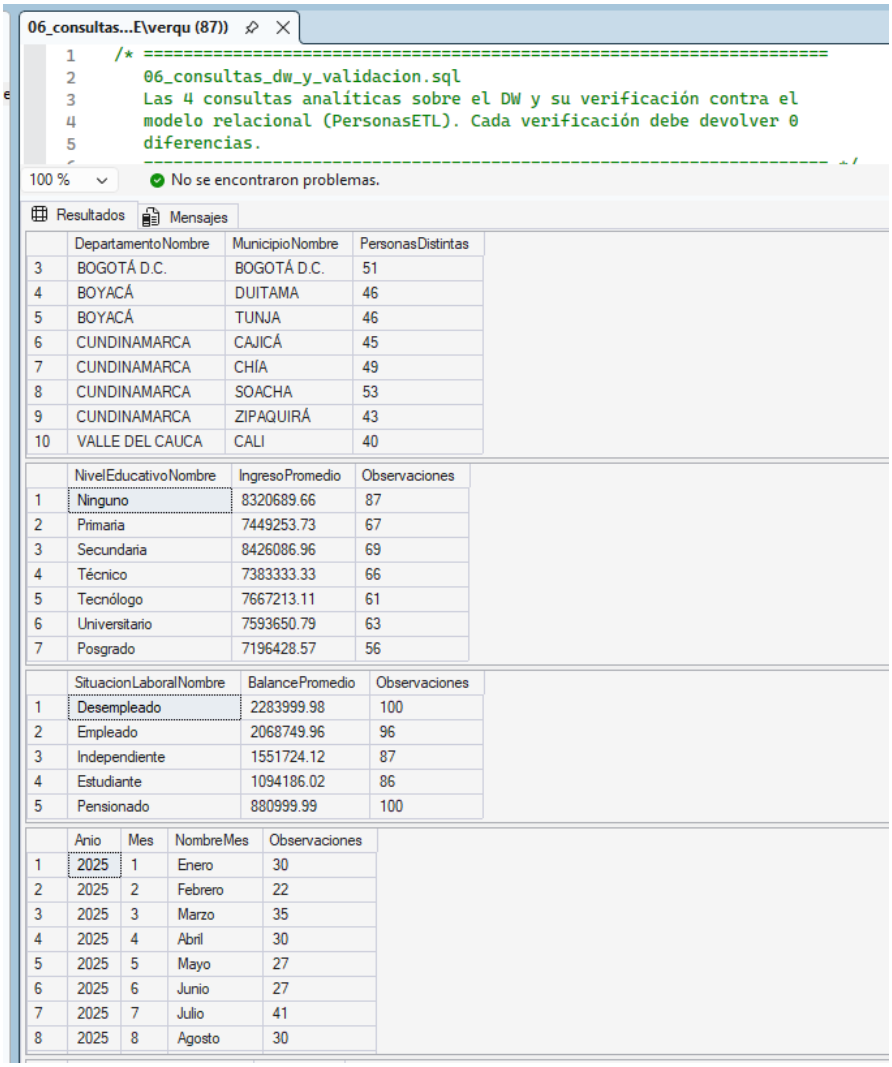
**Tabla 18**

*Cantidad de observaciones por año y mes*

| Año | Mes | Observaciones |
|---|---|---|
| 2025 | 1 | 30 |
| 2025 | 2 | 22 |
| 2025 | 3 | 35 |
| 2025 | 4 | 30 |
| 2025 | 5 | 27 |
| 2025 | 6 | 27 |
| 2025 | 7 | 41 |
| 2025 | 8 | 30 |
| 2026 | 1 | 24 |
| 2026 | 2 | 31 |
| 2026 | 3 | 29 |
| 2026 | 4 | 21 |
| 2026 | 5 | 38 |
| 2026 | 6 | 29 |
| 2026 | 7 | 30 |
| 2026 | 8 | 25 |

**Figura 34**

*Resultados de las cuatro consultas sobre la bodega de datos*




| | DepartamentoNombre | MunicipioNombre | PersonasDistintas |
|---|---|---|---|
| 3 | BOGOTÁ D.C. | BOGOTÁ D.C. | 51 |
| 4 | BOYACÁ | DUITAMA | 46 |
| 5 | BOYACÁ | TUNJA | 46 |
| 6 | CUNDINAMARCA | CAJICÁ | 45 |
| 7 | CUNDINAMARCA | CHÍA | 49 |
| 8 | CUNDINAMARCA | SOACHA | 53 |
| 9 | CUNDINAMARCA | ZIPAQUIRÁ | 43 |
| 10 | VALLE DEL CAUCA | CALI | 40 |

| | NivelEducativoNombre | IngresoPromedio | Observaciones |
|---|---|---|---|
| 1 | Ninguno | 8320689.66 | 87 |
| 2 | Primaria | 7449253.73 | 67 |
| 3 | Secundaria | 8426086.96 | 69 |
| 4 | Técnico | 7383333.33 | 66 |
| 5 | Tecnólogo | 7667213.11 | 61 |
| 6 | Universitario | 7593650.79 | 63 |
| 7 | Posgrado | 7196428.57 | 56 |

| | SituacionLaboralNombre | BalancePromedio | Observaciones |
|---|---|---|---|
| 1 | Desempleado | 2283999.98 | 100 |
| 2 | Empleado | 2068749.96 | 96 |
| 3 | Independiente | 1551724.12 | 87 |
| 4 | Estudiante | 1094186.02 | 86 |
| 5 | Pensionado | 880999.99 | 100 |

| | Anio | Mes | NombreMes | Observaciones |
|---|---|---|---|---|
| 1 | 2025 | 1 | Enero | 30 |
| 2 | 2025 | 2 | Febrero | 22 |
| 3 | 2025 | 3 | Marzo | 35 |
| 4 | 2025 | 4 | Abril | 30 |
| 5 | 2025 | 5 | Mayo | 27 |
| 6 | 2025 | 6 | Junio | 27 |
| 7 | 2025 | 7 | Julio | 41 |
| 8 | 2025 | 8 | Agosto | 30 |

El mismo script recalcula cada consulta en el modelo relacional y cruza ambos resultados con FULL JOIN. Las cuatro verificaciones devolvieron cero diferencias y los totales generales coinciden (Tabla 19 y Figura 35).

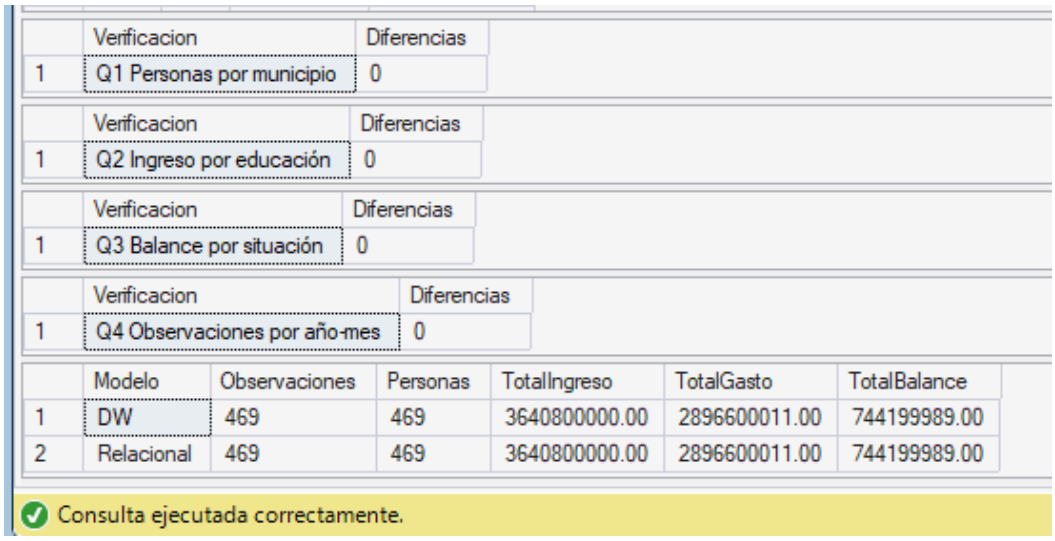
**Tabla 19**

*Totales del modelo relacional y de la bodega de datos*

| Modelo | Observaciones | Personas | Total ingreso | Total gasto | Total balance |
|---|---|---|---|---|---|
| DW | 469 | 469 | 3.640.800.000,00 | 2.896.600.011,00 | 744.199.989,00 |
| Relacional | 469 | 469 | 3.640.800.000,00 | 2.896.600.011,00 | 744.199.989,00 |

**Figura 35**

*Verificación entre la bodega de datos y el modelo relacional*



| | Verificacion | Diferencias |
|---|---|---|
| 1 | Q1 Personas por municipio | 0 |

| | Verificacion | Diferencias |
|---|---|---|
| 1 | Q2 Ingreso por educación | 0 |

| | Verificacion | Diferencias |
|---|---|---|
| 1 | Q3 Balance por situación | 0 |

| | Verificacion | Diferencias |
|---|---|---|
| 1 | Q4 Observaciones por año-mes | 0 |

| | Modelo | Observaciones | Personas | TotalIngreso | TotalGasto | TotalBalance |
|---|---|---|---|---|---|---|
| 1 | DW | 469 | 469 | 3640800000.00 | 2896600011.00 | 744199989.00 |
| 2 | Relacional | 469 | 469 | 3640800000.00 | 2896600011.00 | 744199989.00 |

Consulta ejecutada correctamente.

*Nota.* Cada verificación cuenta las diferencias entre ambos modelos; todas dieron cero.

## Segunda ejecución: prueba de no duplicados

Ambos paquetes se ejecutaron por segunda vez sin borrar el modelo ni la bodega. La ausencia de duplicados se garantiza en tres niveles: las restricciones de unicidad y claves primarias sobre las claves naturales; los Lookups con salida sin coincidencia, que solo dejan pasar filas que aún no existen; y las bitácoras etl.EjecucionPaquete y EjecucionCarga, que registran los conteos de cada corrida (Tabla 20).

**Tabla 20**

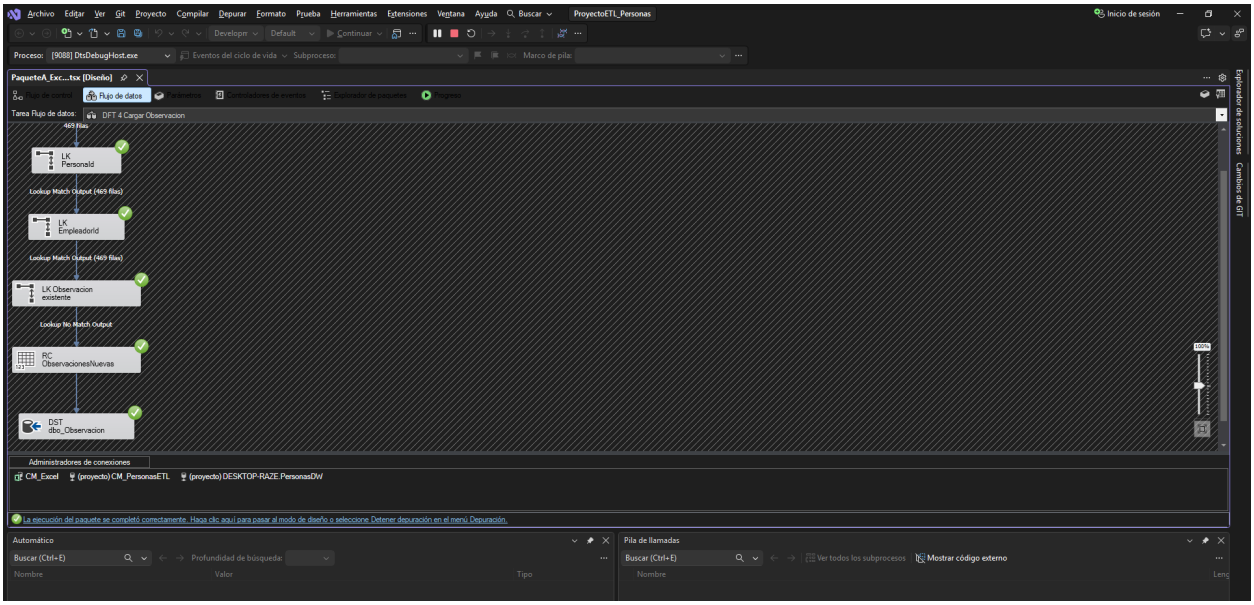
*Filas insertadas en cada ejecución*

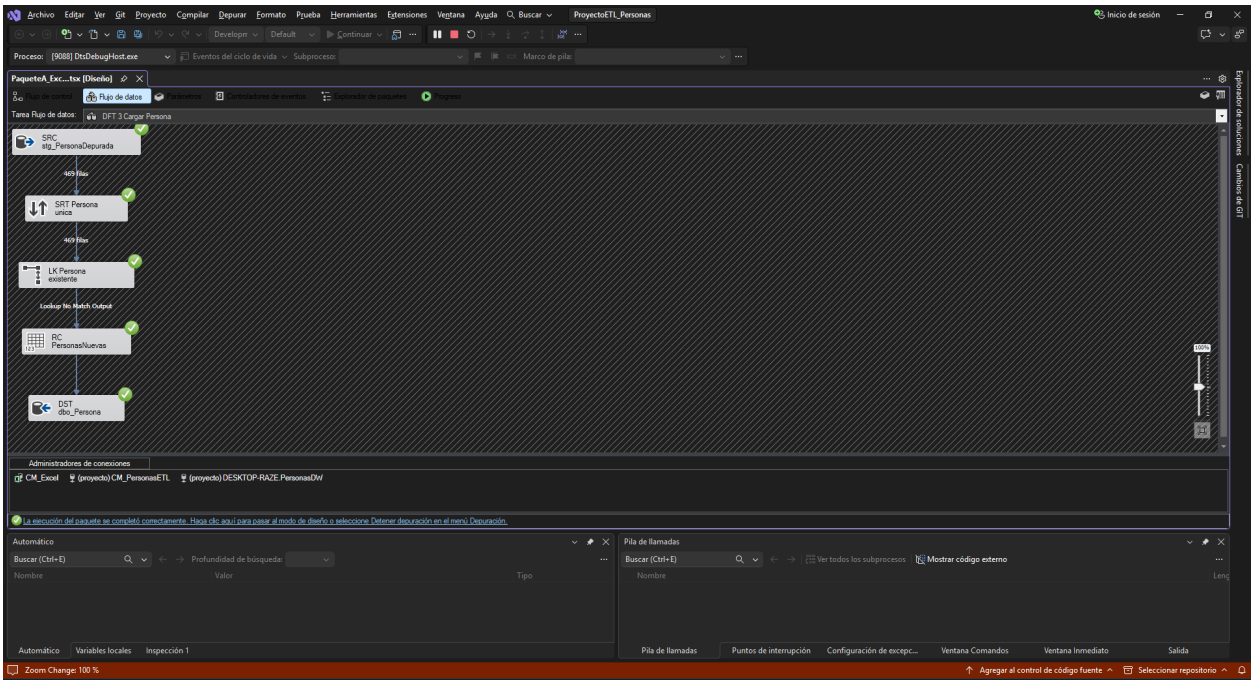
| Ejecución | Personas nuevas | Observaciones nuevas | Dimensiones nuevas | Hechos nuevos |
|---|---|---|---|---|
| Primera | 469 | 469 | 295 + 10 + 7 + 5 + 469 | 469 |
| Segunda | 0 | 0 | 0 | 0 |

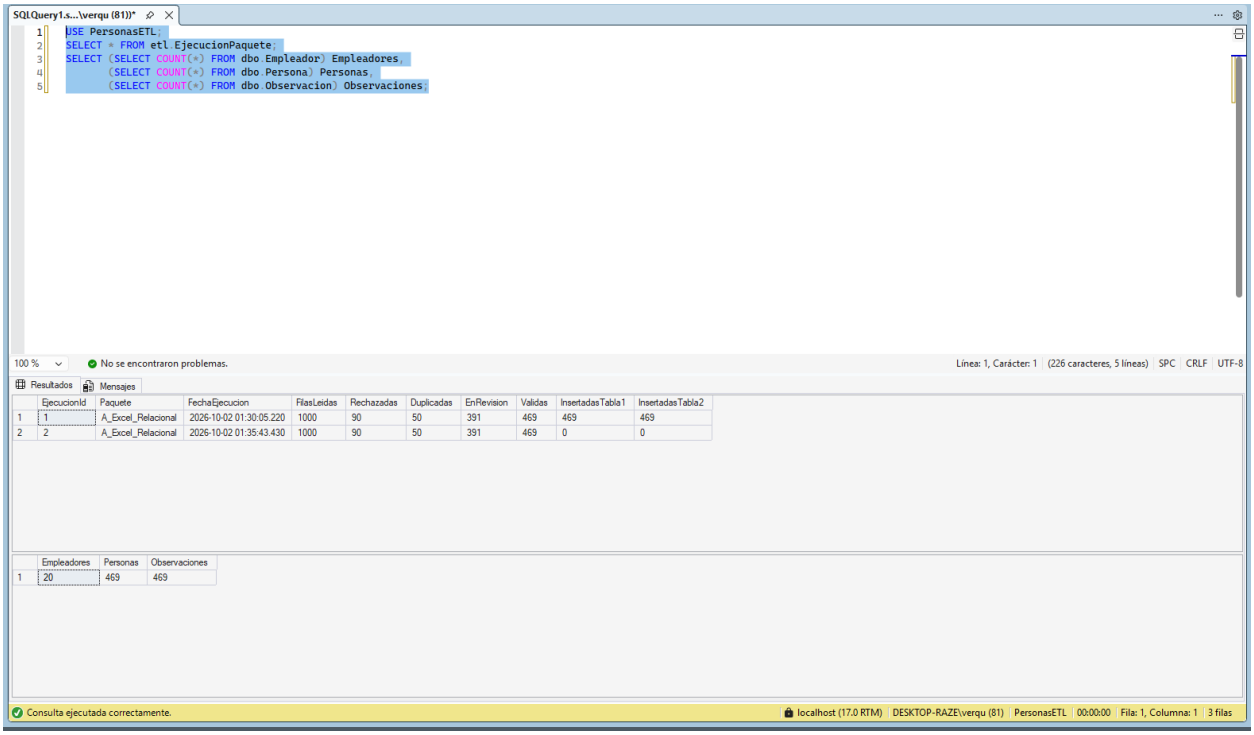
En la segunda ejecución, las salidas sin coincidencia de los flujos de carga no recibieron filas (Figuras 36 y 37), la bitácora del paquete A registró cero personas y cero observaciones nuevas, y las pruebas de duplicados por clave de persona, de observación y de hecho devolvieron cero grupos (Figura 38).

**Figura 36**

*Segunda ejecución del paquete A*



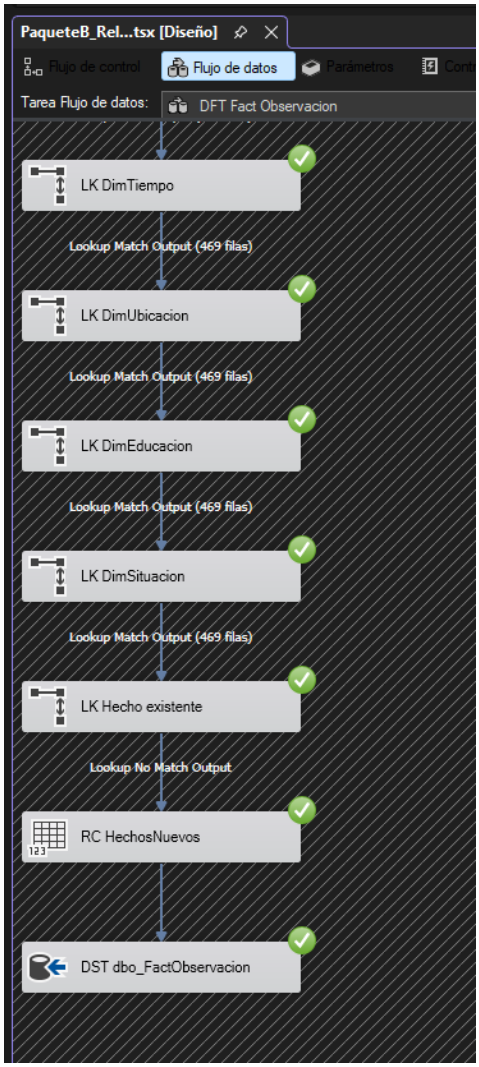



*Nota.* Los flujos DFT 4 y DFT 3 no envían filas a sus destinos, y etl.EjecucionPaquete registra una segunda corrida con cero filas insertadas; los totales siguen en 20 empleadores, 469 personas y 469 observaciones.

**Figura 37**

*Segunda ejecución del flujo de hechos del paquete B*




*Nota.* La salida sin coincidencia de LK Hecho existente no recibe filas, por lo que no se insertan hechos repetidos.

**Figura 38**

*Pruebas de duplicados tras la re-ejecución*

08_evidencia...\verqu (52)) | 06_consultas_...E\verqu (87))

```sql
/* ==================================================================
   08_evidencias_y_reejecucion.sql
   Ejecutar DESPUÉS de correr los paquetes SSIS (y otra vez tras la
   segunda ejecución). Produce las evidencias del informe:
     A. Conteos de aceptados, rechazados, revisión y duplicados
     B. Antes / después de registros de ejemplo
```

100 % | No se encontraron problemas.

Resultados | Mensajes

| | Concepto | Registros |
|---|---|---|
| 1 | Leídas del Excel | 1000 |

| | Tipo | Motivo | Registros |
|---|---|---|---|
| 1 | Rechazo | Documento faltante | 68 |

| | Correccion | Registros |
|---|---|---|
| 1 | Advertencia: personas a cargo > tamaño del hogar | 227 |
| 2 | Teléfono sin guiones | 45 |

| | Momento | SourceRowId | DocumentType | FirstName | LastName | Phone | MonthlyIncome | MonthlyExpenses | Disability |
|---|---|---|---|---|---|---|---|---|---|
| 1 | ANTES | 4 | cc | Laura | López | 000-0000004 | 13900000 | 3700000 | NO |
| 2 | ANTES | 18 | CC | Valentina | Gómez | 0000000018 | $ 14.700.000,00 | 2600000 | SI |
| 3 | ANTES | 22 | CC | Valentina | López | 0000000022 | 5300000 | 8900000,50 | no |

| | Momento | SourceRowId | TipoDocumento | PrimerNombre | PrimerApellido | Telefono | IngresoMensual | GastoMensual | Discapacidad |
|---|---|---|---|---|---|---|---|---|---|
| 1 | DESPUÉS | 4 | CC | Laura | López | 0000000004 | 13900000.00 | 3700000.00 | NO |
| 2 | DESPUÉS | 18 | CC | Valentina | Gómez | 0000000018 | 14700000.00 | 2600000.00 | SI |
| 3 | DESPUÉS | 22 | CC | Valentina | López | 0000000022 | 5300000.00 | 8900000.50 | NO |

| | SourceRowId | Cambios |
|---|---|---|
| 1 | 4 | TipoDocumento a mayúsculas; Teléfono sin guione... |
| 2 | 18 | Ingreso convertido de formato monetario; Adverten... |
| 3 | 22 | Discapacidad estandarizada; Gasto convertido de f... |

| | SourceRowId | TipoDocumento | NumeroDocumento | PrimerNombre | Motivo |
|---|---|---|---|---|---|
| 1 | 2 | PA | NULL | Andrés | Documento faltante; |
| 2 | 33 | PA | SIM0000033 | | Primer nombre faltante; |

| | SourceRowId | NumeroDocumento | BirthDateOriginal | MunicipioCodigo | IngresoOriginal | Motivo |
|---|---|---|---|---|---|---|
| 1 | 5 | SIM0000005 | 31/02/1990 | M006 | 13800000 | Fecha de nacimiento inválida; |
| 2 | 13 | SIM0000013 | 1975-11-10 | M005 | dos millones | Departamento incoherente con municipio; Nombre de departamento incoherente con código; Ingreso no numérico; |

| | Descartado | Conservado | NumeroDocumento | FechaEncuesta | FechaActualizacion | IngresoMensual | Versiones | Motivo |
|---|---|---|---|---|---|---|---|---|
| 1 | 826 | 976 | SIM0000826 | 2025-07-02 | 2025-07-02 12:30:00.000 | 14700000.00 | 2 | Misma persona y fecha de encuesta; se conserva la fila 976 (UpdatedAt más reciente; si empata, la primera aparición) |
| 2 | 951 | 801 | SIM0000801 | 2025-03-27 | 2025-03-27 12:30:00.000 | 2100000.00 | 2 | Misma persona y fecha de encuesta; se conserva la fila 801 (UpdatedAt más reciente; si empata, la primera aparición) |

| | SourceRowId | NumeroDocumento | FechaEncuesta | IngresoMensual | FechaActualizacionOrigen |
|---|---|---|---|---|---|
| 1 | 976 | SIM0000826 | 2025-07-02 | 15200000.00 | 2025-07-02 18:45:00.000 |
| 2 | 801 | SIM0000801 | 2025-03-27 | 2100000.00 | 2025-03-27 12:30:00.000 |

| | Prueba | GruposDuplicados |
|---|---|---|
| 1 | Persona duplicada (tipo+número) | 0 |

Consulta ejecutada correctamente. | localhost (17.0 RTM) | DESKTOP-RAZE\verqu (52) | PersonasETL | 0

*Nota.* En la parte inferior, la prueba de persona duplicada por tipo y número de documento devuelve cero grupos.

## Conclusiones

El cursor procesó los 100 registros solicitados y separó 78 válidos de 22 rechazados, con su motivo. La presencia de 30 registros aceptados con balance negativo confirma que el proceso distinguió entre un dato inválido y un resultado económico desfavorable, sin reemplazar importes desconocidos por cero.

De los 1.000 registros del Excel, 469 llegaron al modelo relacional y a la bodega de datos. Los demás no se perdieron: 90 quedaron en etl.Rechazo, 50 en etl.Duplicado y 391 en etl.Revision, cada uno con su motivo y su SourceRowId. La proporción en revisión es alta porque el archivo contenía errores deliberados en casi todas sus columnas; enviarlos a revisión,

en lugar de completarlos, mantuvo la regla de no inventar información.

El uso de componentes nativos de SSIS fue suficiente para todas las transformaciones, incluida la resolución de duplicados, que se resolvió con Aggregate, Sort, Merge Join y Conditional Split sin recurrir a Script Component. Los Lookups cumplieron un doble papel: validar catálogos y evitar inserciones repetidas.

Las cuatro consultas analíticas de la bodega coincidieron con el modelo relacional, y la segunda ejecución de ambos paquetes no insertó filas nuevas, lo que demuestra que los procesos son correctos y repetibles.

## Repositorio y video de demostración

Los scripts SQL, el proyecto SSIS y la guía de construcción se encuentran en el repositorio https://github.com/Cristof11/Bigdata.

Video de demostración: https://mailunicundiedu-my.sharepoint.com/:f:/g/personal/jcrishtoferrojas_ucundinamarca_edu_co/IgDmpseIN7EkTrw3OSSi0X1wAYaHwb4vUWs2wdqgOY01dtI?e=k7bPAD

## Referencias

American Psychological Association. (2020). *Publication manual of the American Psychological Association* (7.ª ed.). https://doi.org/10.1037/0000165-000

Elmasri, R., y Navathe, S. B. (2016). *Fundamentals of database systems* (7.ª ed.). Pearson.

Kimball, R., y Ross, M. (2013). *The data warehouse toolkit: The definitive guide to dimensional modeling* (3.ª ed.). John Wiley & Sons.

Microsoft. (s.f.-a). *DECLARE CURSOR (Transact-SQL)*. Microsoft Learn. https://learn.microsoft.com/en-us/sql/t-sql/language-elements/declare-cursor-transact-sql

Microsoft. (s.f.-b). *Integration Services transformations*. Microsoft Learn. https://learn.microsoft.com/en-us/sql/integration-services/data-flow/transformations/integration-services-transformations

Microsoft. (s.f.-c). *Lookup transformation*. Microsoft Learn. https://learn.microsoft.com/en-us/sql/integration-services/data-flow/transformations/lookup-transformation

Microsoft. (s.f.-d). *What's new in Integration Services in SQL Server 2025*. Microsoft Learn. https://learn.microsoft.com/en-us/sql/integration-services/what-s-new-in-integration-services-in-sql-server-2025

Rahm, E., y Do, H. H. (2000). Data cleaning: Problems and current approaches. *IEEE Data Engineering Bulletin*, *23*(4), 3–13.