Universidad Nacional Autónoma de México

Facultad de Estudios Superiores Acatlán

Licenciatura en Matemáticas Aplicadas y Computación

Minería de Datos

Semestre 2025-2

Proyecto Final

Por:

López Zubieta Diego Salvador

Ramirez Fernandez Fernando Axel

Vargas González Carlo

Grupo: 2852

Profesor: Javier Rosas Hernández

29-05-2025

Índice

Introducción y Selección del Tópico

Objetivo del Proyecto:

El objetivo de este proyecto es implementar un proceso completo de Extracción, Transformación y Carga (ETL) para procesar, limpiar, almacenar y analizar los datos abiertos de las declaraciones anuales del Impuesto Sobre la Renta (ISR) para Personas Morales, publicados por el Servicio de Administración Tributaria (SAT).

Escenario de Estudio:
Para este análisis, se utilizaron los conjuntos de datos correspondientes a los ejercicios fiscales de **2014 y 2015**, contenidos en los archivos:

- *Anuales_ISR_PM_2014.csv*
- *Anuales_ISR_PM_2015.csv*

Alcance:
Desde la limpieza de datos hasta la creación de un modelo de Data Warehouse y la visualización de insights en un dashboard.

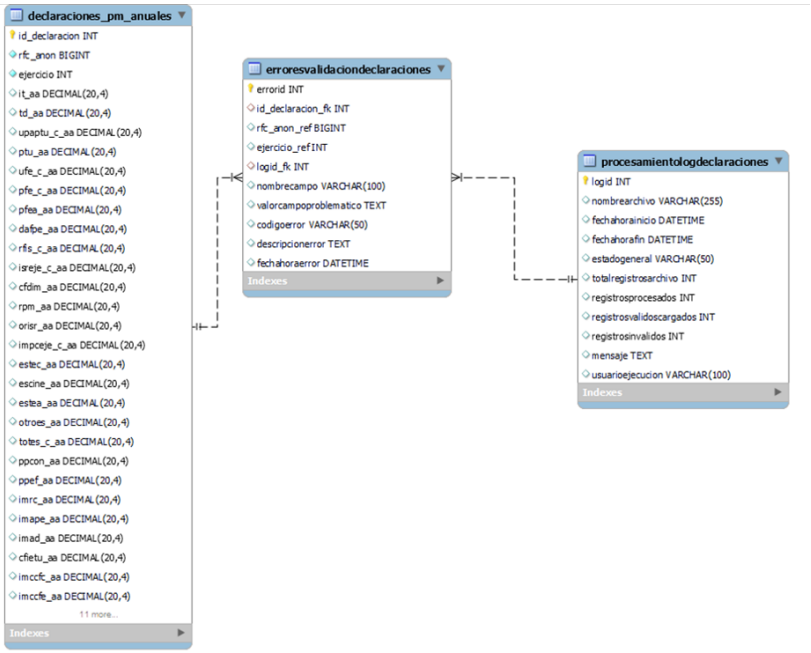
Modelo Transaccional (OLTP)

Propósito:
Se diseñó una base de datos transaccional en MySQL con el propósito de servir como un área de staging o preparación. Su función es almacenar los datos crudos extraídos del CSV después de una primera capa de validación y limpieza, antes de ser transformados y cargados al Data Warehouse.

Estructura de Tablas:

- **declaraciones_pm_anuales**: Tabla principal que almacena cada declaración fiscal.
- **procesamientologdeclaraciones**: Tabla de bitácora para monitorear cada ejecución del ETL, registrando su inicio, fin, estado y conteo de registros.
- **erroresvalidaciondeclaraciones**: Tabla para registrar cada error de validación encontrado, permitiendo un análisis detallado de la calidad de los datos.

Diagrama Entidad-Relación:

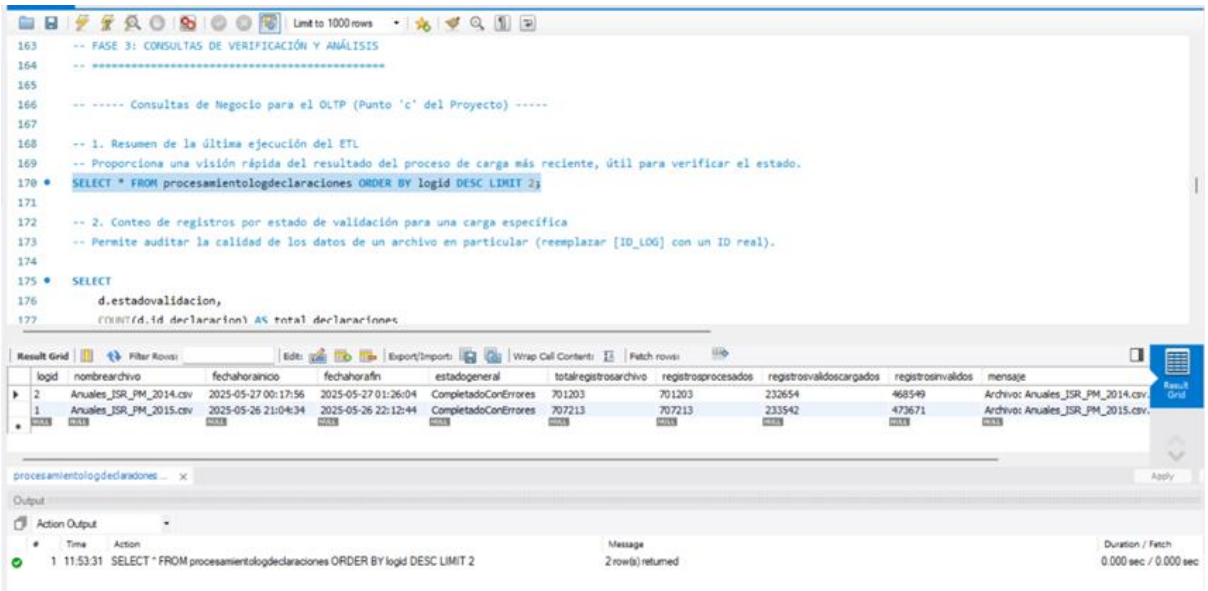


Consultas de Negocio sobre el OLTP

Objetivo de las Consultas:
Estas consultas están diseñadas para auditar el proceso de carga y la calidad de los datos en la fase transaccional, respondiendo a preguntas operativas.

Listado de Consultas:

- **Resumen de la última carga**: Verifica el estado final del proceso ETL más reciente.



- **Conteo de registros por estado**: Cuantifica los registros válidos e inválidos de una carga específica.

```sql
-- 2. Conteo de registros por estado de validación para una carga específica
-- Permite auditar la calidad de los datos de un archivo en particular (reemplazar [ID_LOG] con un ID real).

SELECT
    d.estadovalidacion,
    COUNT(d.id_declaracion) AS total_declaraciones
FROM declaraciones_pm_anuales d
WHERE EXISTS (
    SELECT 1 FROM erroresvalidaciondeclaraciones e
    WHERE e.id_declaracion_fk = d.id_declaracion AND e.logid_fk = 2
)
OR d.estadovalidacion = 'Valido'
GROUP BY d.estadovalidacion;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

estadovalidacion	total_declaraciones
Valido	466196
Invalido	468549

Result 22

Output

Action Output

#	Time	Action	Message
1	11:53:31	SELECT * FROM procesamientologdeclaraciones ORDER BY logid DESC LIMIT 2	2 row(s) returned
2	11:54:19	SELECT d.estadovalidacion, COUNT(d.id_declaracion) AS total_declaraciones FROM declaraciones_p...	2 row(s) returned

- **Frecuencia de errores**: Identifica los problemas de calidad de datos más comunes.

```sql
-- 3. Frecuencia de los tipos de error en la última carga
-- Ayuda a identificar los problemas de calidad de datos más comunes en la fuente de datos más reciente.
SELECT codigoerror, COUNT(*) AS frecuencia
FROM erroresvalidaciondeclaraciones
WHERE logid_fk = (SELECT MAX(logid) FROM procesamientologdeclaraciones)
GROUP BY codigoerror
ORDER BY frecuencia DESC;

-- 4. Muestra de registros con un error específico
-- Facilita el análisis a fondo de un problema particular, mostrando ejemplos concretos.
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

codigoerror	frecuencia
INCONSISTENCIA_UPAPTU	468531
INCONSISTENCIA_TOTES	314

Result 23

Output

Action Output

#	Time	Action	Message
1	11:53:31	SELECT * FROM procesamientologdeclaraciones ORDER BY logid DESC LIMIT 2	2 row(s) returned
2	11:54:19	SELECT d.estadovalidacion, COUNT(d.id_declaracion) AS total_declaraciones FROM declaraciones_p...	2 row(s) returned
3	11:55:57	SELECT codigoerror, COUNT(*) AS frecuencia FROM erroresvalidaciondeclaraciones WHERE logid_fk = (S...	2 row(s) returned

- **Muestra de registros con error**: Permite un análisis detallado de ejemplos de datos problemáticos.

```
194     -- 4. Muestra de registros con un error específico
195     -- Facilita el análisis a fondo de un problema particular, mostrando ejemplos concretos.
196 •   SELECT * FROM declaraciones_pm_anuales
197     WHERE id_declaracion IN (
198         SELECT id_declaracion_fk FROM erroresvalidaciondeclaraciones
199         WHERE codigoerror = 'INCONSISTENCIA_UPAPTU' AND logid_fk = (SELECT MAX(logid) FROM procesamientologdeclaraciones)
200     ) LIMIT 10;
201
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

id_declaracion	rfc_anon	ejercicio	it_aa	td_aa	upaptu_c_aa	ptu_aa	ufe_c_aa	pfe_c_aa	pfea_aa	dafpe_aa	rfs_c_aa	isreje_c_aa	cfdim_aa	rpm_aa	orisr_aa
707214	744964	2014	2374650.7180	2630873.5980	192501.0000	0.0000	29625.2196	NULL	NULL	0.0000	25364.9540	7518.0059	NULL	NULL	NULL
707215	141119	2014	19973180.2500	1538169.4010	19520059.0000	NULL	19639726.0500	NULL	19611648.0400	NULL	0.0000	0.0000	NULL	NULL	NULL
707216	60001	2014	133170.0049	377166.7993	-19814.0000	NULL	0.0000	244640.9224	NULL	NULL	0.0000	0.0000	NULL	NULL	NULL
707217	280696	2014	144893.1328	576516.3669	-224178.0000	NULL	0.0000	409847.4907	NULL	NULL	0.0000	0.0000	NULL	NULL	NULL
707218	493969	2014	14414490.5200	15128843.6700	375482.0000	NULL	368153.1768	NULL	NULL	NULL	453060.0468	140999.4344	NULL	NULL	NULL
707220	936172	2014	10192871.1100	11554131.5800	130454.0000	NULL	125048.2820	NULL	NULL	NULL	117368.3799	36421.6235	NULL	NULL	NULL
707221	724255	2014	409007.8121	620177.7814	25293.0000	NULL	31343.9465	NULL	NULL	NULL	28694.1489	8636.7560	NULL	NULL	NULL
707222	769665	2014	4399775.8850	4729433.5690	0.0000	NULL	200173.6882	NULL	189893.3823	NULL	0.0000	0.0000	NULL	NULL	NULL
707223	144844	2014	13605438.8700	17983917.7100	-3438378.0000	NULL	0.0000	4187676.2310	NULL	NULL	0.0000	0.0000	NULL	NULL	NULL
707225	516912	2014	200039.8498	921751.4698	-99529.0000	NULL	0.0000	199580.0463	NULL	NULL	0.0000	0.0000	NULL	NULL	NULL

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

impceje_c_aa	estec_aa	escine_aa	estea_aa	otroes_aa	totes_c_aa	ppcon_aa	ppef_aa	imrc_aa	imape_aa	imad_aa	cfietu_aa	imccfc_aa	imccfe_aa	occ_aa	ocf_aa	difcar_c_aa	diffav_c_aa	im
6182.0471	NULL	NULL	NULL	3907.7725	3444.5312	0.0000	0.0000	0.0000	0.0000	0.0000	NULL	NULL	NULL	NULL	NULL	0.0000	5569.0712	NULL
0.0000	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0.0000	0.0000	NULL
0.0000	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0.0000	0.0000	NULL
0.0000	NULL	NULL	NULL	NULL	NULL	NULL	287.3258	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0.0000	0.0000	NULL
140999.4344	NULL	NULL	NULL	NULL	NULL	NULL	211474.7200	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0.0000	89794.6808	NULL
36421.6235	NULL	NULL	NULL	NULL	NULL	NULL	71711.9531	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0.0000	41493.5732	NULL
8636.7560	NULL	NULL	NULL	NULL	NULL	NULL	1903.0540	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	6495.9754	0.0000	NULL
0.0000	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0.0000	0.0000	NULL
0.0000	NULL	NULL	NULL	NULL	NULL	NULL	95302.9362	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0.0000	115547.3216	NULL
0.0000	NULL	NULL	NULL	NULL	NULL	NULL	3307.8335	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0.0000	693.2761	NULL

_aa	imape_aa	imad_aa	cfietu_aa	imccfc_aa	imccfe_aa	occ_aa	ocf_aa	difcar_c_aa	diffav_c_aa	imide_aa	impsun_aa	isrcareje_c_aa	isrpeimpac_aa	isrfaveje_c_aa	fechacarga	estadovalidacion
000	0.0000	0.0000	NULL	NULL	NULL	NULL	NULL	0.0000	5569.0712	NULL	NULL	NULL	NULL	5569.0712	2025-05-27 00:17:56	Invalido
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0.0000	0.0000	NULL	NULL	0.0000	NULL	NULL	2025-05-27 00:17:56	Invalido
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0.0000	0.0000	NULL	NULL	0.0000	NULL	NULL	2025-05-27 00:17:56	Invalido
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0.0000	0.0000	NULL	NULL	NULL	NULL	0.0000	2025-05-27 00:17:56	Invalido
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0.0000	89794.6808	NULL	NULL	NULL	NULL	89794.6808	2025-05-27 00:17:56	Invalido
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0.0000	41493.5732	NULL	NULL	NULL	NULL	41493.5732	2025-05-27 00:17:56	Invalido
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	6495.9754	0.0000	NULL	NULL	6495.9754	NULL	NULL	2025-05-27 00:17:56	Invalido
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0.0000	0.0000	NULL	NULL	0.0000	NULL	NULL	2025-05-27 00:17:56	Invalido
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0.0000	115547.3216	NULL	NULL	NULL	NULL	115547.3216	2025-05-27 00:17:56	Invalido
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0.0000	693.2761	NULL	NULL	0.0000	NULL	693.2761	2025-05-27 00:17:56	Invalido

- **Verificación de duplicados**: Confirma la integridad de los datos y la correcta aplicación de la restricción UNIQUE en la base de datos.

OLTP*

Limit to 1000 rows

```
202     -- 5. Verificación de duplicados cargados en la tabla OLTP
203     -- Es una consulta operativa clave para garantizar la integridad de los datos y que la constraint UNIQUE está funcionando.
204 •   SELECT rfc_anon, ejercicio, COUNT(*) AS numero_de_registros
205     FROM declaraciones_pm_anuales
206     GROUP BY rfc_anon, ejercicio
207     HAVING COUNT(*) > 1;
208
209
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

rfc_anon	ejercicio	numero_de_registros

Proceso ETL: De CSV a OLTP

Descripción del Script:

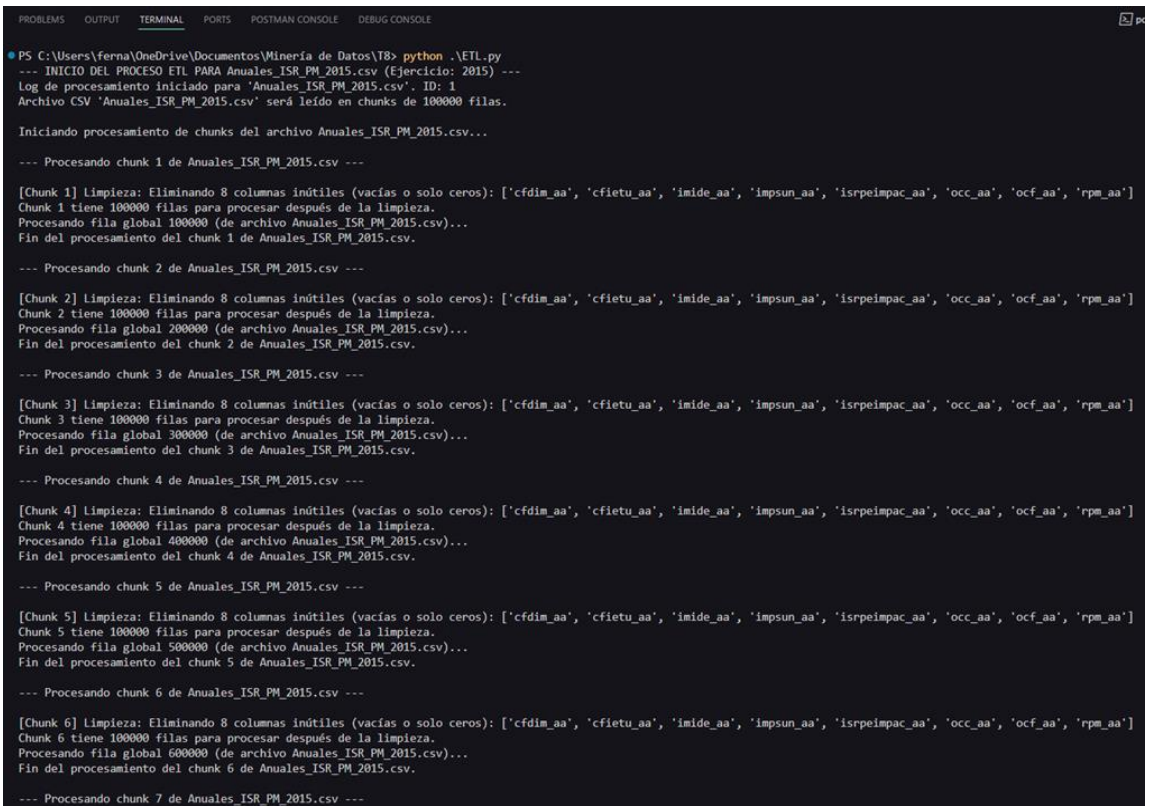
Este script realiza la “primera fase del proceso ETL”. Su principal responsabilidad es tomar los datos crudos del archivo CSV y cargarlos en una base de datos transaccional (OLTP)

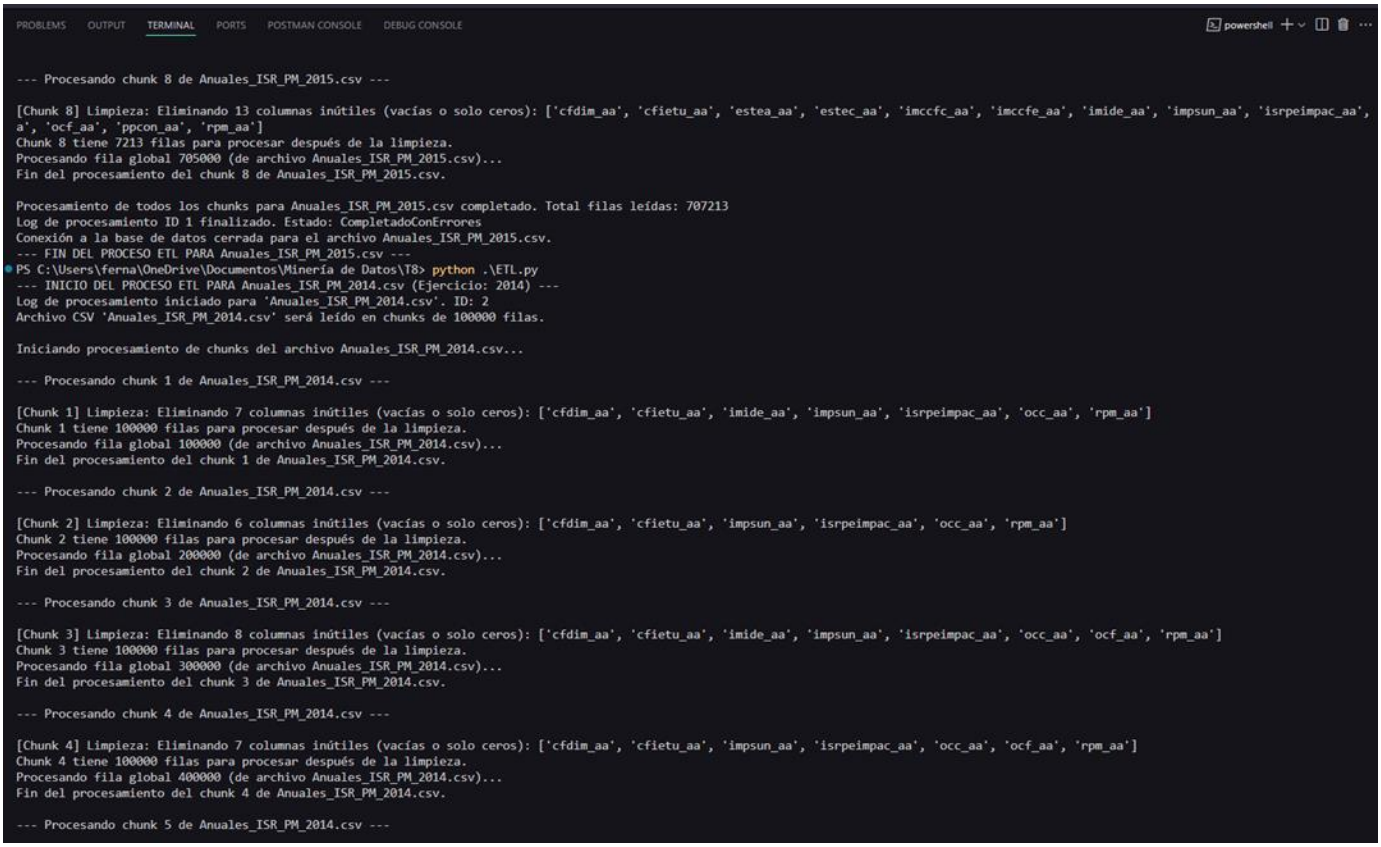
que sirve como área de preparación o staging. Durante este proceso, el script realiza tres tareas críticas:

1. **Limpia los datos dinámicamente**, eliminando sobre la marcha cualquier columna que no aporte información útil (aquellas que están completamente vacías o llenas de ceros).
2. **Valida cada fila** contra un conjunto de reglas de negocio para asegurar su calidad y consistencia.
3. **Carga los registros** en la tabla principal, asignando un estado de 'Valido' o 'Invalido' y registrando cualquier error específico en una tabla de auditoría para su posterior análisis.

Fases del Proceso:

- **Extracción**: Se leen los archivos CSV por lotes (chunks) de 100,000 filas para un manejo eficiente de la memoria.
- **Transformación y Limpieza:**
    - **Limpieza Dinámica de Columnas**: Se implementó una función limpiar_columnas_inutiles que, en cada chunk, identifica y elimina automáticamente las columnas que contienen únicamente valores nulos o ceros, optimizando el almacenamiento y el procesamiento.
- **Validación de Datos**: Cada fila es sometida a un riguroso proceso de validación que incluye: comprobación de campos obligatorios, valores no negativos, valores esperados (como el ejercicio fiscal) y reglas de consistencia cruzada (ej. upaptu_c_aa = it_aa - td_aa).
- **Carga**: Los registros se insertan en la tabla declaraciones_pm_anuales, asignando un estadovalidacion de 'Valido' o 'Invalido'. Los errores específicos se registran en erroresvalidaciondeclaraciones.





Modelo del Data Warehouse (DW)

Arquitectura: Para el análisis de datos, se diseñó e implementó un Data Warehouse con un Esquema de Estrella (Star Schema), utilizando MySQL como sistema gestor de bases de datos.
Componentes del Modelo:

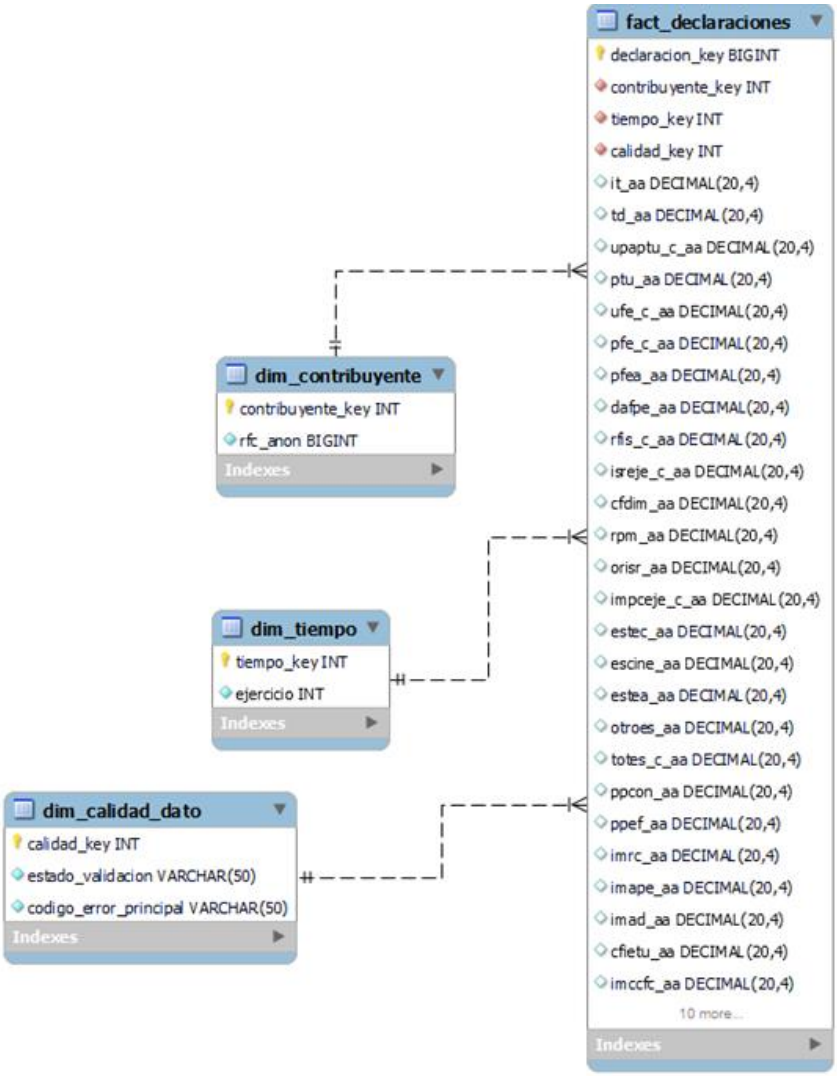
Tabla de Hechos (fact_declaraciones): Contiene las métricas numéricas y las claves foráneas.
Dimensiones:

- dim_tiempo: Describe el eje de tiempo (cuándo).
- dim_contribuyente: Describe a los contribuyentes (quién).
- dim_calidad_dato: Clasifica la calidad de cada registro (qué tipo).

```
PS C:\Users\ferna\OneDrive\Documentos\Minería de Datos\T8> python .\ETL_DWH.py
¡Conexión exitosa a la base de datos MySQL!
Limpiando tablas del Data Warehouse...
Tablas del DW limpiadas exitosamente.
Poblando dim_tiempo...
dim_tiempo poblada con 2 nuevo(s) registro(s).
Poblando dim_contribuyente...
dim_contribuyente poblada con 785691 registros.
Poblando dim_calidad_dato...
dim_calidad_dato poblada con 4 categorías.

Iniciando población de fact_declaraciones...
Cargando dimensiones en memoria...
C:\Users\ferna\OneDrive\Documentos\Minería de Datos\T8\ETL_DWH.py:107: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connectio
. Other DBAPI2 objects are not tested. Please consider using SQLAlchemy.
  tiempo_map = pd.read_sql("SELECT tiempo_key, ejercicio FROM dim_tiempo", conn).set_index('ejercicio')['tiempo_key']
C:\Users\ferna\OneDrive\Documentos\Minería de Datos\T8\ETL_DWH.py:108: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connectio
. Other DBAPI2 objects are not tested. Please consider using SQLAlchemy.
  contrib_map = pd.read_sql("SELECT contribuyente_key, rfc_anon FROM dim_contribuyente", conn).set_index('rfc_anon')['contribuyente_key']
C:\Users\ferna\OneDrive\Documentos\Minería de Datos\T8\ETL_DWH.py:109: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connectio
. Other DBAPI2 objects are not tested. Please consider using SQLAlchemy.
  calidad_df = pd.read_sql("SELECT calidad_key, estado_validacion, codigo_error_principal FROM dim_calidad_dato", conn)
Dimensiones cargadas.
Leyendo todos los datos de declaraciones_pm_anuales para fact_declaraciones...
C:\Users\ferna\OneDrive\Documentos\Minería de Datos\T8\ETL_DWH.py:140: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connectio
. Other DBAPI2 objects are not tested. Please consider using SQLAlchemy.
  df_oltp_completo_original = pd.read_sql(sql_extract, conn)
Se leyeron 1408416 filas de la tabla OLTP.
Procesando el DataFrame completo para la tabla de hechos...

--- INFO: Preparando tuplas y convirtiendo NaN a None ---
Intentando insertar lote de 10000 filas (total insertadas hasta ahora: 0)...
Lote insertado. 10000 de 1408416 filas insertadas en total en fact_declaraciones.
Intentando insertar lote de 10000 filas (total insertadas hasta ahora: 10000)...
Lote insertado. 20000 de 1408416 filas insertadas en total en fact_declaraciones.
Intentando insertar lote de 10000 filas (total insertadas hasta ahora: 20000)...
Lote insertado. 30000 de 1408416 filas insertadas en total en fact_declaraciones.
Intentando insertar lote de 10000 filas (total insertadas hasta ahora: 30000)...
Lote insertado. 40000 de 1408416 filas insertadas en total en fact_declaraciones.
Intentando insertar lote de 10000 filas (total insertadas hasta ahora: 40000)...
Lote insertado. 50000 de 1408416 filas insertadas en total en fact_declaraciones.
Intentando insertar lote de 10000 filas (total insertadas hasta ahora: 50000)...
Lote insertado. 60000 de 1408416 filas insertadas en total en fact_declaraciones.
Intentando insertar lote de 10000 filas (total insertadas hasta ahora: 60000)...
Lote insertado. 70000 de 1408416 filas insertadas en total en fact_declaraciones.
Intentando insertar lote de 10000 filas (total insertadas hasta ahora: 70000)...
Lote insertado. 80000 de 1408416 filas insertadas en total en fact_declaraciones.
Intentando insertar lote de 10000 filas (total insertadas hasta ahora: 80000)...
Lote insertado. 90000 de 1408416 filas insertadas en total en fact_declaraciones.
```

```
Lote insertado. 1350000 de 1408416 filas insertadas en total en fact_declaraciones.
Intentando insertar lote de 10000 filas (total insertadas hasta ahora: 1350000)...
Lote insertado. 1360000 de 1408416 filas insertadas en total en fact_declaraciones.
Intentando insertar lote de 10000 filas (total insertadas hasta ahora: 1360000)...
Lote insertado. 1370000 de 1408416 filas insertadas en total en fact_declaraciones.
Intentando insertar lote de 10000 filas (total insertadas hasta ahora: 1370000)...
Lote insertado. 1380000 de 1408416 filas insertadas en total en fact_declaraciones.
Intentando insertar lote de 10000 filas (total insertadas hasta ahora: 1380000)...
Lote insertado. 1390000 de 1408416 filas insertadas en total en fact_declaraciones.
Intentando insertar lote de 10000 filas (total insertadas hasta ahora: 1390000)...
Lote insertado. 1400000 de 1408416 filas insertadas en total en fact_declaraciones.
Intentando insertar lote de 8416 filas (total insertadas hasta ahora: 1400000)...
Lote insertado. 1408416 de 1408416 filas insertadas en total en fact_declaraciones.

Proceso de población de fact_declaraciones finalizado en 170.15 segundos.

¡Proceso ETL de OLTP a DW finalizado!
Conexión a la base de datos cerrada.
PS C:\Users\ferna\OneDrive\Documentos\Minería de Datos\T8>
```

Diagrama del Modelo:



Proceso ETL: De OLTP a DW

Descripción del Script: ETL_DWH.py (Carga de OLTP a DW)

Este script ejecuta la “segunda y última fase del ETL”, transformando los datos ya validados del sistema OLTP en un modelo analítico optimizado para inteligencia de negocio. Su objetivo es construir el Data Warehouse (DW) con una estructura de Esquema de Estrella. Para ello:

1. **Puebla las tablas de dimensiones** (dim_tiempo, dim_contribuyente, dim_calidad_dato) con los atributos descriptivos del negocio.
2. **Construye la tabla de hecho**s (fact_declaraciones) central, llenándola con las métricas cuantitativas y vinculándola a las dimensiones a través de sus claves.

En esencia, este script convierte la lista plana de datos en un modelo multidimensional listo para ser explotado eficientemente por herramientas de visualización como Power BI.

Fases del Proceso:

- **Extracción:** Se extraen todos los datos limpios de la tabla declaraciones_pm_anuales del sistema OLTP.
- **Transformación y Carga:**
    - Se pueblan primero las tablas de dimensiones (dim_tiempo, dim_contribuyente, dim_calidad_dato) obteniendo los valores únicos del OLTP.
    - Posteriormente, se construye la tabla de hechos (fact_declaraciones) mapeando las claves de negocio (ej. rfc_anon, ejercicio) a las claves subrogadas (ej. contribuyente_key, tiempo_key) de las dimensiones.

```
217     -- Muestra de datos de la tabla de hechos con dimensiones
218 •   SELECT
219         fd.declaracion_key,
220         dc.rfc_anon,
221         dt.ejercicio,
222         dqd.estado_validacion,
223         dqd.codigo_error_principal,
224         fd.it_aa,
225         fd.td_aa,
226         fd.upaptu_c_aa
227     FROM fact_declaraciones fd
228     JOIN dim_contribuyente dc ON fd.contribuyente_key = dc.contribuyente_key
229     JOIN dim_tiempo dt ON fd.tiempo_key = dt.tiempo_key
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

declaracion_key	rfc_anon	ejercicio	estado_validacion	codigo_error_principal	it_aa	td_aa	upaptu_c_aa
707214	744964	2014	Invalido	INCONSISTENCIA_UPAPTU	2374650.7180	2630873.5980	192501.0000
707215	141119	2014	Invalido	INCONSISTENCIA_UPAPTU	19973180.2500	1538169.4010	19520059.0000
707216	60001	2014	Invalido	INCONSISTENCIA_UPAPTU	133170.0049	377166.7993	-19814.0000
707217	280696	2014	Invalido	INCONSISTENCIA_UPAPTU	144893.1328	576516.3669	-224178.0000
707218	493969	2014	Invalido	INCONSISTENCIA_UPAPTU	14414490.5200	15128843.6700	375482.0000
707219	79196	2014	Valido	NINGUNO	0.0000	NULL	0.0000
707220	936172	2014	Invalido	INCONSISTENCIA_UPAPTU	10192871.1100	11554131.5800	130454.0000
707221	724255	2014	Invalido	INCONSISTENCIA_UPAPTU	409007.8121	620177.7814	25293.0000
707222	769665	2014	Invalido	INCONSISTENCIA_UPAPTU	4399775.8850	4729433.5690	0.0000
707223	144844	2014	Invalido	INCONSISTENCIA_UPAPTU	13605438.8700	17983917.7100	-3438378.0000

dim_tiempo 31 | Result 32 | Result 33

```
210     -- ----- Consultas de Verificación del DW (Después de ETL_DWH.py) -----
211     -- Verificar contenido de dim_tiempo
212 •   SELECT * FROM dim_tiempo ORDER BY ejercicio;
213
214     -- Contar el total de hechos en fact_declaraciones
215 •   SELECT COUNT(*) AS total_hechos FROM fact_declaraciones;
216
217     -- Muestra de datos de la tabla de hechos con dimensiones
218 •   SELECT
219         fd.declaracion_key,
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

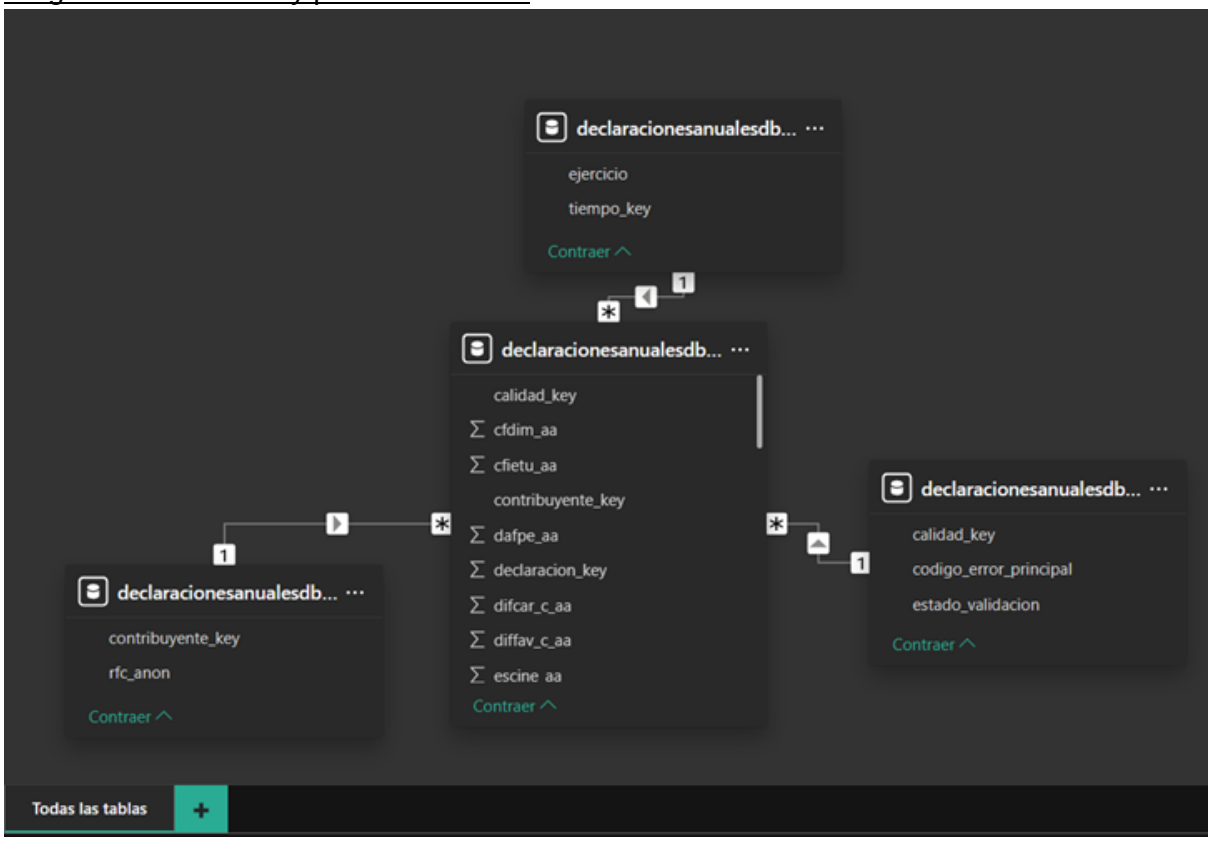
tiempo_key	ejercicio
1	2014
2	2015
NULL	NULL

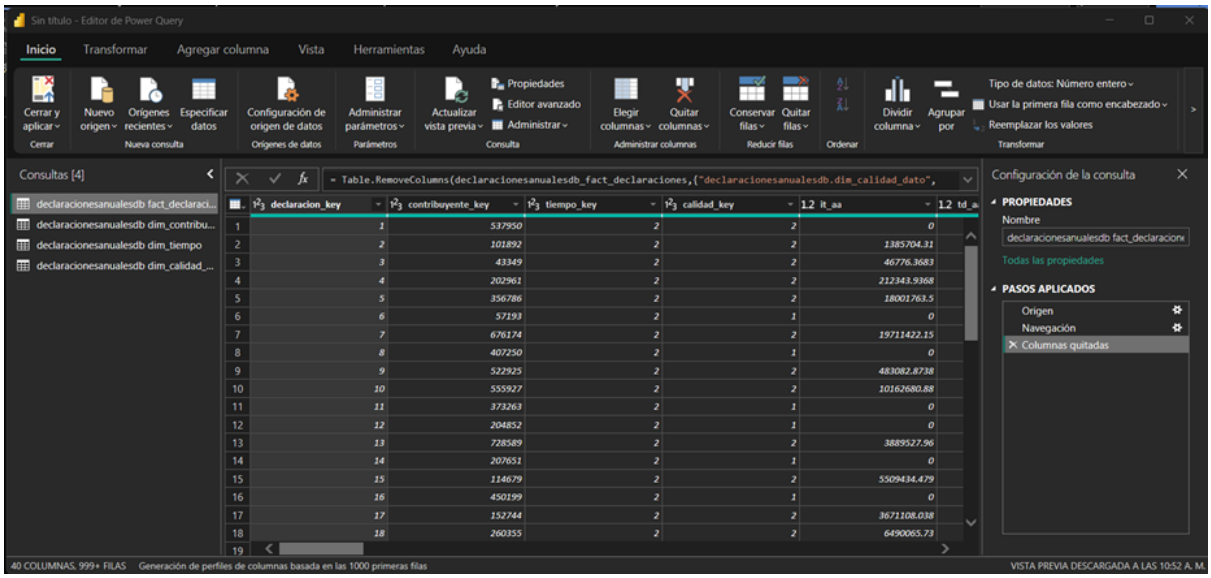
```
210     -- ----- Consultas de Verificación del DW (Después de ETL_DWH.py) -----
211     -- Verificar contenido de dim_tiempo
212 •   SELECT * FROM dim_tiempo ORDER BY ejercicio;
213
214     -- Contar el total de hechos en fact_declaraciones
215 •   SELECT COUNT(*) AS total_hechos FROM fact_declaraciones;
216
217     -- Muestra de datos de la tabla de hechos con dimensiones
218 •   SELECT
219         fd.declaracion_key,
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total_hechos
1408416

Diagrama del Modelo y proceso de ETL:





Consultas de Negocio sobre el DW

Objetivo de las Consultas: Estas consultas están diseñadas para realizar análisis de inteligencia de negocio, identificando tendencias, patrones y comparativas a lo largo del tiempo.

Listado de Consultas:

- **Ingresos y Resultado Fiscal por Año**: Permite un análisis macro de la rentabilidad año contra año.

```sql
-- 1. Ingresos Acumulables Totales (IT_AA) y Resultado Fiscal Total (RFIS_c_AA) por Año:
SELECT
    t.ejercicio,
    SUM(f.it_aa) AS TotalIngresosAcumulables,
    SUM(f.rfis_c_aa) AS TotalResultadoFiscal
FROM fact_declaraciones f
JOIN dim_tiempo t ON f.tiempo_key = t.tiempo_key
GROUP BY t.ejercicio
ORDER BY t.ejercicio;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ejercicio	TotalIngresosAcumulables	TotalResultadoFiscal
2014	16931682789433.0619	834173841674.8404
2015	20863147978115.0181	1206507261780.9648

- **Distribución de Declaraciones por Calidad**: Mide el impacto de la calidad de los datos en el conjunto total de registros.

```sql
-- 2. Distribución de Declaraciones por Estado de Validación y Código de Error Principal, por Año:
SELECT
    t.ejercicio,
    qd.estado_validacion,
    qd.codigo_error_principal,
    COUNT(f.declaracion_key) AS NumeroDeDeclaraciones
FROM fact_declaraciones f
JOIN dim_tiempo t ON f.tiempo_key = t.tiempo_key
JOIN dim_calidad_dato qd ON f.calidad_key = qd.calidad_key
GROUP BY t.ejercicio, qd.estado_validacion, qd.codigo_error_principal
ORDER BY t.ejercicio, NumeroDeDeclaraciones DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ejercicio	estado_validacion	codigo_error_principal	NumeroDeDeclaraciones
2014	Invalido	INCONSISTENCIA_UPAPTU	468531
2014	Valido	NINGUNO	232654
2014	Invalido	INCONSISTENCIA_TOTES	18
2015	Invalido	INCONSISTENCIA_UPAPTU	473663
2015	Valido	NINGUNO	233542
2015	Invalido	INCONSISTENCIA_TOTES	8

- **Comparativa de ISR**: Analiza cómo una inconsistencia específica afecta a una métrica fiscal clave.

```sql
-- 3. Comparativa Anual del ISR del Ejercicio (ISREJE_c_AA) para Declaraciones "Válidas" vs. "INCONSISTENCIA_UPAPTU":
SELECT
    t.ejercicio,
    qd.codigo_error_principal,
    AVG(f.isreje_c_aa) AS Promedio_ISR_Ejercicio,
    COUNT(f.declaracion_key) AS NumeroDeDeclaraciones
FROM fact_declaraciones f
JOIN dim_tiempo t ON f.tiempo_key = t.tiempo_key
JOIN dim_calidad_dato qd ON f.calidad_key = qd.calidad_key
WHERE qd.codigo_error_principal IN ('NINGUNO', 'INCONSISTENCIA_UPAPTU')
GROUP BY t.ejercicio, qd.codigo_error_principal
ORDER BY t.ejercicio, qd.codigo_error_principal;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ejercicio	codigo_error_principal	Promedio_ISR_Ejercicio	NumeroDeDeclaraciones
2014	INCONSISTENCIA_UPAPTU	527368.11748624	468531
2014	NINGUNO	4711.28107543	232654
2015	INCONSISTENCIA_UPAPTU	757681.04957210	473663
2015	NINGUNO	14363.68203612	233542

- **Top 5 Contribuyentes por Pérdida**: Identifica outliers o casos de estudio importantes.

```sql
-- 4. Top 5 RFCs por Suma de Pérdidas Fiscales del Ejercicio (PFE_c_AA) en 2015:
SELECT
    c.rfc_anon,
    SUM(f.pfe_c_aa) AS TotalPerdidaFiscal_2015
FROM fact_declaraciones f
JOIN dim_contribuyente c ON f.contribuyente_key = c.contribuyente_key
JOIN dim_tiempo t ON f.tiempo_key = t.tiempo_key
WHERE t.ejercicio = 2015 AND f.pfe_c_aa > 0
GROUP BY c.rfc_anon
ORDER BY TotalPerdidaFiscal_2015 DESC
LIMIT 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

rfc_anon	TotalPerdidaFiscal_2015
1012538	128016000000.0000
825826	32040375480.0000
992718	7062367523.0000
982120	4497997804.0000
38640	3953440167.0000

- **Evolución del PTU:** Muestra tendencias en el reparto de utilidades a los trabajadores.

```sql
-- 5. Evolución del Promedio de PTU Pagada (PTU_AA) por Año (solo para quienes pagaron PTU):
SELECT
    t.ejercicio,
    AVG(f.ptu_aa) AS Promedio_PTU_Pagada,
    COUNT(DISTINCT f.contribuyente_key) AS NumeroDeEmpresasQuePagaronPTU
FROM fact_declaraciones f
JOIN dim_tiempo t ON f.tiempo_key = t.tiempo_key
WHERE f.ptu_aa > 0
GROUP BY t.ejercicio
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

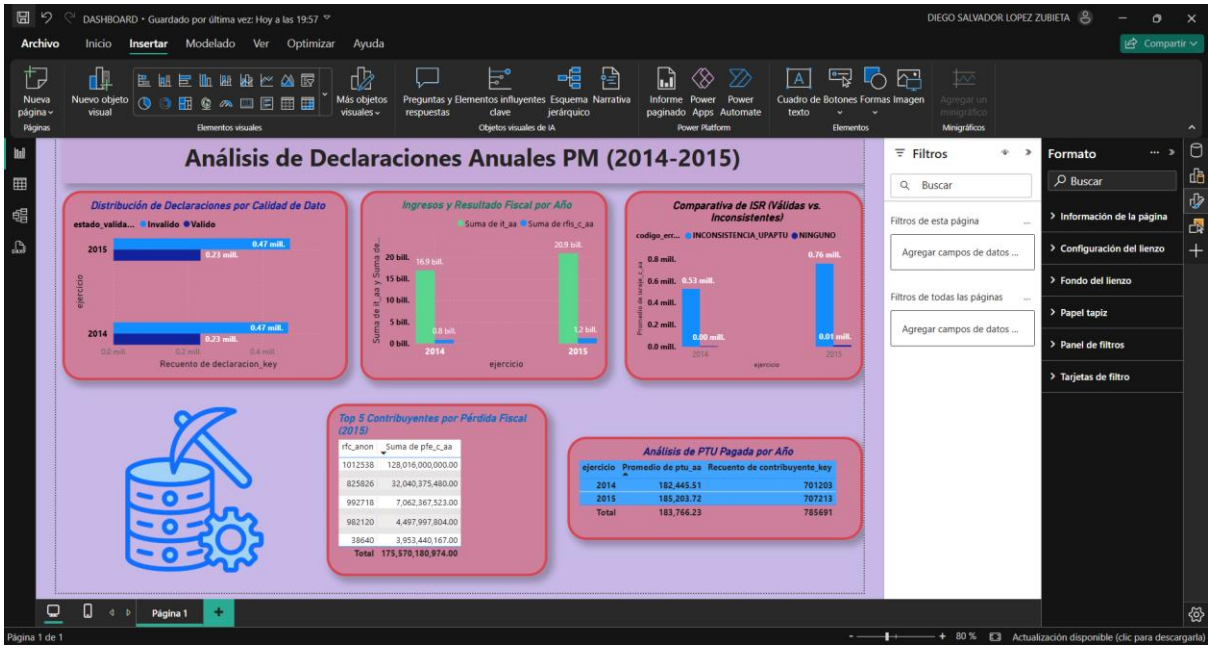
ejercicio	Promedio_PTU_Pagada	NumeroDeEmpresasQuePagaronPTU
2014	862040.88227044	49048
2015	676078.66070376	58328

Dashboard y Visualización de Datos

Herramienta: Se utilizó Microsoft Power BI para la creación de un dashboard interactivo.

Objetivo del Dashboard: Presentar los hallazgos de las consultas de negocio del DW de una manera visual, intuitiva y accesible para un usuario no técnico.

<u>Descripción de Visuales:</u>



Descripción de los Componentes del Dashboard

- **Ingresos y Resultado Fiscal por Año**: Este gráfico de columnas verticales compara dos métricas financieras clave para los años 2014 y 2015: la Suma de Ingresos Acumulables (it_aa) y la Suma del Resultado Fiscal (rfis_c_aa). Respondiendo a la primera consulta de negocio, permite una evaluación rápida del desempeño financiero general y su variación interanual.

- **Distribución de Declaraciones por Calidad de Dato**: Mediante un gráfico de barras horizontales, esta visualización muestra el número total de declaraciones procesadas cada año, segmentado por su estado de validación ('Válido' o 'Inválido'). Este componente aborda la segunda consulta de negocio, ofreciendo una perspectiva clara sobre la calidad general de los datos y el volumen de registros que presentaron inconsistencias en cada periodo.
- **Comparativa de ISR (Válidas vs. Inconsistentes)**: Este gráfico de columnas ilustra el impacto de la calidad de los datos en una métrica crítica. Compara el Promedio del ISR del Ejercicio (isreje_c_aa) entre las declaraciones consideradas válidas (NINGUNO) y aquellas con el error más común (INCONSISTENCIA_UPAPTU). Responde a la tercera pregunta de negocio, evidenciando la diferencia significativa en los valores reportados.
- **Top 5 Contribuyentes por Pérdida Fiscal (2015)**: Presentado en formato de tabla, este visual identifica los cinco contribuyentes (rfc_anon) con la mayor Suma de Pérdida Fiscal (pfe_c_aa) durante el ejercicio 2015. Da respuesta a la cuarta consulta de negocio, permitiendo un análisis enfocado en los casos más extremos o atípicos del conjunto de datos.
- **Análisis de PTU Pagada por Año**: Esta tabla final resume el comportamiento del Reparto de Utilidades (PTU). Muestra, para 2014 y 2015, el Promedio de PTU pagado por las empresas y el Recuento de contribuyentes que realizaron dicho pago. Con esto, se responde a la quinta pregunta de negocio sobre la evolución de esta prestación a los trabajadores.

Conclusiones:

En este proyecto, construimos con éxito un sistema completo para procesar los datos de las declaraciones de impuestos de 2014 y 2015. Creamos los scripts de Python que leen los archivos CSV, los limpian y los cargan en una base de datos organizada, lista para el análisis. El proyecto culminó con la creación de un dashboard en Power BI que muestra los resultados de forma visual.

El mayor aprendizaje fue que los datos reales no vienen limpios. Encontramos muchos problemas como registros duplicados, columnas vacías y errores en los cálculos. Aprendimos a solucionar estos problemas usando herramientas específicas: la base de datos se configuró para rechazar duplicados automáticamente, y el código de Python se programó para detectar y limpiar las columnas inútiles y validar la información antes de guardarla.

El resultado final es un dashboard que transforma datos complicados en gráficos fáciles de entender, permitiendo analizar la información fiscal de manera rápida. Cómo siguientes pasos, este sistema podría mejorarse agregando datos de más años para ver tendencias más largas, o incluso se podría intentar crear un modelo que prediga qué declaraciones podrían tener errores en el futuro.

Anexos

Anexo A: Código fuente de ETL.py.

```
import pandas as pd
import mysql.connector
from mysql.connector import errorcode
import datetime
import decimal
import os # Para obtener solo el nombre del archivo

# --- Configuración de Archivo y Ejercicio ---
# CAMBIA ESTOS VALORES SEGÚN EL ARCHIVO QUE SE QUIERAS PROCESAR
# Para procesar 2014:
CSV_FILE_TO_PROCESS = 'Anuales_ISR_PM_2014.csv' # Asegúrate que este
archivo exista
EJERCICIO_FISCAL_ESPERADO = 2014

# Para procesar 2015 (descomenta estas líneas y comenta las de 2014):
#CSV_FILE_TO_PROCESS = 'Anuales_ISR_PM_2015.csv'
#EJERCICIO_FISCAL_ESPERADO = 2015
# --- Fin de Configuración ---

# --- Configuración de la Base de Datos ---
DB_CONFIG = {
    'host': '127.0.0.1',
    'port': 3306,
    'user': 'root',
    'password': 'root',
    'database': 'DeclaracionesAnualesDB'
}

# --- Metadatos de Columnas (con 'Variable' en minúsculas) ---
column_metadata = [
    {'Variable': 'rfc_anon', 'Descripción': 'Identificador del RFC',
'tipo_dato_original': 'BIGINT', 'tipo_dato_pandas': 'Int64',
'tipo_dato_sql': 'BIGINT', 'obligatorio': True},
    {'Variable': 'ejercicio', 'Descripción': 'Ejercicio Fiscal
Presentado', 'tipo_dato_original': 'INTEGER', 'tipo_dato_pandas':
'Int64', 'tipo_dato_sql': 'INT', 'obligatorio': True, 'valor_esperado':
EJERCICIO_FISCAL_ESPERADO},
    {'Variable': 'it_aa', 'Descripción': 'TOTAL DE INGRESOS
ACUMULABLES', 'tipo_dato_original': 'FLOAT', 'tipo_dato_pandas':
'float64', 'tipo_dato_sql': 'DECIMAL(20, 4)', 'no_negativo': True},
```

```
    {'Variable': 'td_aa', 'Descripción': 'TOTAL DE DEDUCCIONES
AUTORIZADAS Y DEDUCCIÓN INMEDIATA DE INVERSIONES',
'tipo_dato_original': 'FLOAT', 'tipo_dato_pandas': 'float64',
'tipo_dato_sql': 'DECIMAL(20, 4)', 'no_negativo': True},
    {'Variable': 'upaptu_c_aa', 'Descripción': 'UTILIDAD O PERDIDA
FISCAL ANTES DE PTU', 'tipo_dato_original': 'FLOAT',
'tipo_dato_pandas': 'float64', 'tipo_dato_sql': 'DECIMAL(20, 4)'},
    {'Variable': 'ptu_aa', 'Descripción': 'PTU PAGADA EN EL EJERCICIO',
'tipo_dato_original': 'FLOAT', 'tipo_dato_pandas': 'float64',
'tipo_dato_sql': 'DECIMAL(20, 4)', 'no_negativo': True},
    {'Variable': 'ufe_c_aa', 'Descripción': 'UTILIDAD FISCAL DEL
EJERCICIO', 'tipo_dato_original': 'FLOAT', 'tipo_dato_pandas':
'float64', 'tipo_dato_sql': 'DECIMAL(20, 4)'},
    {'Variable': 'pfe_c_aa', 'Descripción': 'PERDIDA FISCAL DEL
EJERCICIO', 'tipo_dato_original': 'FLOAT', 'tipo_dato_pandas':
'float64', 'tipo_dato_sql': 'DECIMAL(20, 4)', 'no_negativo': True},
    {'Variable': 'pfea_aa', 'Descripción': 'PERDIDAS FISCALES DE
EJERCICIOS ANTERIORES QUE SE APLICAN EN EL EJERCICIO',
'tipo_dato_original': 'FLOAT', 'tipo_dato_pandas': 'float64',
'tipo_dato_sql': 'DECIMAL(20, 4)', 'no_negativo': True},
    {'Variable': 'dafpe_aa', 'Descripción': 'DEDUCCIÓN ADICIONAL DEL
FOMENTO AL PRIMER EMPLEO', 'tipo_dato_original': 'FLOAT',
'tipo_dato_pandas': 'float64', 'tipo_dato_sql': 'DECIMAL(20, 4)',
'no_negativo': True},
    {'Variable': 'rfis_c_aa', 'Descripción': 'RESULTADO FISCAL',
'tipo_dato_original': 'FLOAT', 'tipo_dato_pandas': 'float64',
'tipo_dato_sql': 'DECIMAL(20, 4)'},
    {'Variable': 'isreje_c_aa', 'Descripción': 'IMPUESTO SOBRE LA RENTA
DEL EJERCICIO', 'tipo_dato_original': 'FLOAT', 'tipo_dato_pandas':
'float64', 'tipo_dato_sql': 'DECIMAL(20, 4)', 'no_negativo': True},
    {'Variable': 'cfdim_aa', 'Descripción': 'REDUCCIONES PARA
MAQUILADORAS', 'tipo_dato_original': 'FLOAT', 'tipo_dato_pandas':
'float64', 'tipo_dato_sql': 'DECIMAL(20, 4)', 'no_negativo': True},
    {'Variable': 'rpm_aa', 'Descripción': 'CREDITO FISCAL POR DEDUCCION
INMEDIATA PARA MAQUILADORAS', 'tipo_dato_original': 'FLOAT',
'tipo_dato_pandas': 'float64', 'tipo_dato_sql': 'DECIMAL(20, 4)',
'no_negativo': True},
    {'Variable': 'orisr_aa', 'Descripción': 'OTRAS REDUCCIONES DEL
ISR', 'tipo_dato_original': 'FLOAT', 'tipo_dato_pandas': 'float64',
'tipo_dato_sql': 'DECIMAL(20, 4)', 'no_negativo': True},
    {'Variable': 'impceje_c_aa', 'Descripción': 'IMPUESTO CAUSADO EN EL
EJERCICIO', 'tipo_dato_original': 'FLOAT', 'tipo_dato_pandas':
'float64', 'tipo_dato_sql': 'DECIMAL(20, 4)', 'no_negativo': True},
```

```
    {'Variable': 'estec_aa', 'Descripción': 'ESTIMULO POR PROYECTOS EN
INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO', 'tipo_dato_original': 'FLOAT',
'tipo_dato_pandas': 'float64', 'tipo_dato_sql': 'DECIMAL(20, 4)',
'no_negativo': True},
    {'Variable': 'escine_aa', 'Descripción': 'ESTIMULO POR PROYECTOS DE
INVERSIÓN EN LA PRODUCCIÓN CINEMATOGRÁFICA NACIONAL',
'tipo_dato_original': 'FLOAT', 'tipo_dato_pandas': 'float64',
'tipo_dato_sql': 'DECIMAL(20, 4)', 'no_negativo': True},
    {'Variable': 'estea_aa', 'Descripción': 'ESTIMULO A PROYECTOS DE
INVERSIÓN EN LA PRODUCCIÓN TEATRAL NACIONAL', 'tipo_dato_original':
'FLOAT', 'tipo_dato_pandas': 'float64', 'tipo_dato_sql': 'DECIMAL(20,
4)', 'no_negativo': True},
    {'Variable': 'otroes_aa', 'Descripción': 'OTROS ESTÍMULOS',
'tipo_dato_original': 'FLOAT', 'tipo_dato_pandas': 'float64',
'tipo_dato_sql': 'DECIMAL(20, 4)', 'no_negativo': True},
    {'Variable': 'totes_c_aa', 'Descripción': 'TOTAL DE ESTIMULOS',
'tipo_dato_original': 'FLOAT', 'tipo_dato_pandas': 'float64',
'tipo_dato_sql': 'DECIMAL(20, 4)', 'no_negativo': True},
    {'Variable': 'ppcon_aa', 'Descripción': 'PAGOS PROVISIONALES
EFECTUADOS ENTREGADOS A LA CONTROLADORA', 'tipo_dato_original':
'FLOAT', 'tipo_dato_pandas': 'float64', 'tipo_dato_sql': 'DECIMAL(20,
4)', 'no_negativo': True},
    {'Variable': 'ppef_aa', 'Descripción': 'PAGOS PROVISIONIALES
EFECTUADOS ENTERADOS A LA FEDERACIÓN', 'tipo_dato_original': 'FLOAT',
'tipo_dato_pandas': 'float64', 'tipo_dato_sql': 'DECIMAL(20, 4)',
'no_negativo': True},
    {'Variable': 'imrc_aa', 'Descripción': 'IMPUESTO RETENIDO AL
CONTRIBUYENTE', 'tipo_dato_original': 'FLOAT', 'tipo_dato_pandas':
'float64', 'tipo_dato_sql': 'DECIMAL(20, 4)', 'no_negativo': True},
    {'Variable': 'imape_aa', 'Descripción': 'IMPUESTO ACREDITARLE
PAGADO EN EL EXTRANJERO', 'tipo_dato_original': 'FLOAT',
'tipo_dato_pandas': 'float64', 'tipo_dato_sql': 'DECIMAL(20, 4)',
'no_negativo': True},
    {'Variable': 'imad_aa', 'Descripción': 'IMPUESTO ACREDITARLE POR
D1V1DENDOS O UTILIDADES DISTRIBUIDOS', 'tipo_dato_original': 'FLOAT',
'tipo_dato_pandas': 'float64', 'tipo_dato_sql': 'DECIMAL(20, 4)',
'no_negativo': True},
    {'Variable': 'cfietu_aa', 'Descripción': 'CRÉDITO FISCAL IETU POR
DEDUCCIONES MAYORES A LOS INGRESOS', 'tipo_dato_original': 'FLOAT',
'tipo_dato_pandas': 'float64', 'tipo_dato_sql': 'DECIMAL(20, 4)',
'no_negativo': True},
    {'Variable': 'imccfc_aa', 'Descripción': 'IMPUESTO CORRESPONDIENTE
A LA CONSOLIDACIÓN FISCAL A CARGO', 'tipo_dato_original': 'FLOAT',
```

```
'tipo_dato_pandas': 'float64', 'tipo_dato_sql': 'DECIMAL(20, 4)',
'no_negativo': True},
    {'Variable': 'imccfe_aa', 'Descripción': 'IMPUESTO CORRESPONDIENTE
A LA CONSOLIDACION FISCAL ENTREGADO (en exceso)', 'tipo_dato_original':
'FLOAT', 'tipo_dato_pandas': 'float64', 'tipo_dato_sql': 'DECIMAL(20,
4)', 'no_negativo': True},
    {'Variable': 'occ_aa', 'Descripción': 'OTRAS CANTIDADES A CARGO',
'tipo_dato_original': 'FLOAT', 'tipo_dato_pandas': 'float64',
'tipo_dato_sql': 'DECIMAL(20, 4)', 'no_negativo': True},
    {'Variable': 'ocf_aa', 'Descripción': 'OTRAS CANTIDADES A FAVOR',
'tipo_dato_original': 'FLOAT', 'tipo_dato_pandas': 'float64',
'tipo_dato_sql': 'DECIMAL(20, 4)', 'no_negativo': True},
    {'Variable': 'difcar_c_aa', 'Descripción': 'DIFERENCIA A CARGO',
'tipo_dato_original': 'FLOAT', 'tipo_dato_pandas': 'float64',
'tipo_dato_sql': 'DECIMAL(20, 4)', 'no_negativo': True},
    {'Variable': 'diffav_c_aa', 'Descripción': 'DIFERENCIA A FAVOR',
'tipo_dato_original': 'FLOAT', 'tipo_dato_pandas': 'float64',
'tipo_dato_sql': 'DECIMAL(20, 4)', 'no_negativo': True},
    {'Variable': 'imide_aa', 'Descripción': 'IMPUESTO ACREDITARLE POR
DEPÓSITOS EN EFECTIVO DEL EJERCICIO', 'tipo_dato_original': 'FLOAT',
'tipo_dato_pandas': 'float64', 'tipo_dato_sql': 'DECIMAL(20, 4)',
'no_negativo': True},
    {'Variable': 'impsun_aa', 'Descripción': 'IMPUESTO ALA VENTA DE
BIENES Y SERVICIOS SUNTUARIOS ACREDITARLE', 'tipo_dato_original':
'FLOAT', 'tipo_dato_pandas': 'float64', 'tipo_dato_sql': 'DECIMAL(20,
4)', 'no_negativo': True},
    {'Variable': 'isrcareje_c_aa', 'Descripción': 'ISR A CARGO DEL
EJERCICIO', 'tipo_dato_original': 'FLOAT', 'tipo_dato_pandas':
'float64', 'tipo_dato_sql': 'DECIMAL(20, 4)', 'no_negativo': True},
    {'Variable': 'isrpeimpac_aa', 'Descripción': 'ISR PAGADO EN EXCESO
APLICADO CONTRA EL IMPAC', 'tipo_dato_original': 'FLOAT',
'tipo_dato_pandas': 'float64', 'tipo_dato_sql': 'DECIMAL(20, 4)',
'no_negativo': True},
    {'Variable': 'isrfaveje_c_aa', 'Descripción': 'ISR A FAVOR DEL
EJERCICIO', 'tipo_dato_original': 'FLOAT', 'tipo_dato_pandas':
'float64', 'tipo_dato_sql': 'DECIMAL(20, 4)', 'no_negativo': True}
]

CHUNK_SIZE = 100000

def get_db_connection():
    try:
        conn = mysql.connector.connect(**DB_CONFIG)
        return conn
```

```
    except mysql.connector.Error as err:
        if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
            print("Error: Usuario o contraseña de MySQL incorrectos.")
        elif err.errno == errorcode.ER_BAD_DB_ERROR:
            print(f"Error: La base de datos '{DB_CONFIG['database']}'
no existe.")
        else:
            print(f"Error de conexión a la base de datos: {err}")
        return None

def preprocess_chunk(chunk_df, metadata):
    for col_meta in metadata:
        col_name = col_meta['Variable']
        if col_name in chunk_df.columns:
            if col_meta['tipo_dato_original'] == 'BIGINT' or
col_meta['tipo_dato_original'] == 'INTEGER':
                chunk_df[col_name] = pd.to_numeric(chunk_df[col_name],
errors='coerce')
                if not chunk_df[col_name].isnull().all():
                    is_integer_like = (chunk_df[col_name].dropna() % 1
== 0).all()
                    if is_integer_like:
                        chunk_df[col_name] =
chunk_df[col_name].astype(col_meta['tipo_dato_pandas'])
            elif col_meta['tipo_dato_original'] == 'FLOAT':
                chunk_df[col_name] = pd.to_numeric(chunk_df[col_name],
errors='coerce').astype(col_meta['tipo_dato_pandas'])
    return chunk_df

def read_csv_in_chunks(file_path, chunk_size_param):
    try:
        if not os.path.exists(file_path):
            print(f"Error: Archivo CSV '{file_path}' no encontrado.")
            return None
        print(f"Archivo CSV '{file_path}' será leído en chunks de
{chunk_size_param} filas.")
        return pd.read_csv(file_path, delimiter=',',
chunksize=chunk_size_param, low_memory=False)
    except Exception as e:
        print(f"Error al leer el archivo CSV en chunks: {e}")
        return None

def iniciar_log_procesamiento(conn, nombre_archivo_completo):
    cursor = conn.cursor()
```

```
    usuario_db = "etl_script"
    try:
        cursor.execute("SELECT USER()")
        usuario_db_result = cursor.fetchone()
        if usuario_db_result:
            usuario_db = usuario_db_result[0]
    except Exception as e:
        print(f"Advertencia: No se pudo obtener el usuario de BD,
usando default '{usuario_db}'. Error: {e}")

    sql_insert_log = """
    INSERT INTO procesamientologdeclaraciones
    (nombrearchivo, fechahorainicio, estadogeneral, usuarioejecucion)
    VALUES (%s, %s, %s, %s)
    """
    ahora = datetime.datetime.now()
    nombre_archivo_base = os.path.basename(nombre_archivo_completo)
    valores = (nombre_archivo_base, ahora, 'Iniciado', usuario_db)

    try:
        cursor.execute(sql_insert_log, valores)
        conn.commit()
        log_id = cursor.lastrowid
        print(f"Log de procesamiento iniciado para
'{nombre_archivo_base}'. ID: {log_id}")
        return log_id
    except mysql.connector.Error as err:
        print(f"Error al iniciar log de procesamiento para
'{nombre_archivo_base}': {err}")
        conn.rollback()
        return None
    finally:
        cursor.close()

def finalizar_log_procesamiento(conn, log_id, estado, total_arch, proc,
validos, invalidos, msg=""):
    if not log_id: return
    cursor = conn.cursor()
    sql_update_log = """
    UPDATE procesamientologdeclaraciones SET
    fechahorafin = %s, estadogeneral = %s, totalregistrosarchivo = %s,
    registrosprocesados = %s, registrosvalidoscargados = %s,
    registrosinvalidos = %s, mensaje = %s
    WHERE logid = %s
```

```
    """
    ahora = datetime.datetime.now()
    valores = (ahora, estado, total_arch, proc, validos, invalidos,
msg, log_id)
    try:
        cursor.execute(sql_update_log, valores)
        conn.commit()
        print(f"Log de procesamiento ID {log_id} finalizado. Estado:
{estado}")
    except mysql.connector.Error as err:
        print(f"Error al finalizar log de procesamiento ID {log_id}:
{err}")
        conn.rollback()
    finally:
        cursor.close()

def registrar_error_validacion(conn, log_id, rfc_ref, ejercicio_ref,
campo, valor, codigo, descripcion, id_declaracion_fk=None):
    cursor = conn.cursor()
    sql_insert_error = """
    INSERT INTO erroresvalidaciondeclaraciones
    (id_declaracion_fk, rfc_anon_ref, ejercicio_ref, logid_fk,
nombrecampo, valorcampoproblematico, codigoerror, descripcionerror)
    VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
    """
    valor_str = str(valor) if valor is not None else None
    rfc_ref_clean = None if pd.isna(rfc_ref) else rfc_ref
    ejercicio_ref_clean = None if pd.isna(ejercicio_ref) else
int(ejercicio_ref) if pd.notna(ejercicio_ref) else None

    valores = (id_declaracion_fk, rfc_ref_clean, ejercicio_ref_clean,
log_id, campo, valor_str, codigo, descripcion)
    try:
        cursor.execute(sql_insert_error, valores)
        conn.commit()
    except mysql.connector.Error as err:
        print(f"\nError al registrar error de validación para RFC
{rfc_ref_clean}, Campo {campo}: {err}")
        conn.rollback()
    finally:
        cursor.close()

def validar_fila(row_series, metadata_list, log_id, conn):
    errores_en_fila = []
```

```
    rfc_anon = row_series.get('rfc_anon')
    ejercicio = row_series.get('ejercicio')

    for col_meta in metadata_list:
        col_name = col_meta['Variable']
        valor = row_series.get(col_name)

        if col_meta.get('obligatorio') and pd.isna(valor):
            desc = f"El campo '{col_name}' es obligatorio y está
vacío."
            errores_en_fila.append({'campo': col_name, 'valor': valor,
'codigo': 'OBLIGATORIO_FALTANTE', 'desc': desc})
            if conn: registrar_error_validacion(conn, log_id, rfc_anon,
ejercicio, col_name, valor, 'OBLIGATORIO_FALTANTE', desc)
            continue

        if pd.isna(valor):
            continue

        valor_esperado = col_meta.get('valor_esperado')
        if valor_esperado is not None:
            try:
                tipo_esperado = type(valor_esperado)
                valor_convertido_para_comparacion =
tipo_esperado(valor)
                if valor_convertido_para_comparacion != valor_esperado:
                    desc = f"El campo '{col_name}' tiene un valor
'{valor}' pero se esperaba '{valor_esperado}'."
                    errores_en_fila.append({'campo': col_name, 'valor':
valor, 'codigo': 'VALOR_INESPERADO', 'desc': desc})
                    if conn: registrar_error_validacion(conn, log_id,
rfc_anon, ejercicio, col_name, valor, 'VALOR_INESPERADO', desc)
            except ValueError:
                desc = f"El campo '{col_name}' tiene un valor '{valor}'
que no se pudo convertir al tipo esperado para la comparación con
'{valor_esperado}'."
                errores_en_fila.append({'campo': col_name, 'valor':
valor, 'codigo': 'TIPO_DATO_INESPERADO', 'desc': desc})
                if conn: registrar_error_validacion(conn, log_id,
rfc_anon, ejercicio, col_name, valor, 'TIPO_DATO_INESPERADO', desc)

        if col_meta.get('no_negativo') and isinstance(valor, (int,
float, decimal.Decimal)) and valor < 0:
```

```
            desc = f"El campo '{col_name}' tiene un valor negativo
'{valor}' y no está permitido."
            errores_en_fila.append({'campo': col_name, 'valor': valor,
'codigo': 'NEGATIVO_NO_PERMITIDO', 'desc': desc})
            if conn: registrar_error_validacion(conn, log_id, rfc_anon,
ejercicio, col_name, valor, 'NEGATIVO_NO_PERMITIDO', desc)

        if col_name == 'upaptu_c_aa':
            it_aa_val = row_series.get('it_aa')
            td_aa_val = row_series.get('td_aa')
            if pd.notna(it_aa_val) and pd.notna(td_aa_val) and
pd.notna(valor):
                try:
                    calculo_upaptu = decimal.Decimal(str(it_aa_val)) -
decimal.Decimal(str(td_aa_val))
                    diferencia = abs(calculo_upaptu -
decimal.Decimal(str(valor)))
                    if diferencia > decimal.Decimal('0.01'):
                        desc_cruzada = (f"Inconsistencia UPAPTU: it_aa
({it_aa_val}) - td_aa ({td_aa_val}) = {calculo_upaptu}, "
                                        f"pero upaptu_c_aa reportado es
{valor}. Diferencia: {diferencia:.4f}")
                        errores_en_fila.append({'campo':
'UPAPTU_C_AA_CROSSCHECK', 'valor': valor,
                                                'codigo':
'INCONSISTENCIA_UPAPTU', 'desc': desc_cruzada})
                        if conn: registrar_error_validacion(conn,
log_id, rfc_anon, ejercicio,

'UPAPTU_C_AA_CROSSCHECK',
f"IT_AA:{it_aa_val},TD_AA:{td_aa_val},UPAPTU_C_AA:{valor}",

'INCONSISTENCIA_UPAPTU', desc_cruzada)
                except decimal.InvalidOperation: pass

        if col_name == 'totes_c_aa':
            estec_aa_val = row_series.get('estec_aa')
            escine_aa_val = row_series.get('escine_aa')
            estea_aa_val = row_series.get('estea_aa')
            otroes_aa_val = row_series.get('otroes_aa')
            if pd.notna(valor):
                try:
```

```
                    suma_estimulos =
(decimal.Decimal(str(estec_aa_val)) if pd.notna(estec_aa_val) else
decimal.Decimal(0)) + \

(decimal.Decimal(str(escine_aa_val)) if pd.notna(escine_aa_val) else
decimal.Decimal(0)) + \

(decimal.Decimal(str(estea_aa_val)) if pd.notna(estea_aa_val) else
decimal.Decimal(0)) + \

(decimal.Decimal(str(otroes_aa_val)) if pd.notna(otroes_aa_val) else
decimal.Decimal(0))
                    diferencia_totes = abs(suma_estimulos -
decimal.Decimal(str(valor)))
                    if diferencia_totes > decimal.Decimal('0.01'):
                        desc_cruzada = (f"Inconsistencia TOTES: Suma de
estímulos individuales ({suma_estimulos}) "
                                        f"no coincide con totes_c_aa
({valor}). Diferencia: {diferencia_totes:.4f}")
                        errores_en_fila.append({'campo':
'TOTES_C_AA_CROSSCHECK', 'valor': valor,
                                                'codigo':
'INCONSISTENCIA_TOTES', 'desc': desc_cruzada})
                        if conn: registrar_error_validacion(conn,
log_id, rfc_anon, ejercicio,

'TOTES_C_AA_CROSSCHECK',
f"SumaInd:{suma_estimulos},TOTES_C_AA:{valor}",

'INCONSISTENCIA_TOTES', desc_cruzada)
                except decimal.InvalidOperation: pass
    return errores_en_fila

def insertar_declaracion(conn, data_dict, estado_validacion,
metadata_list):
    cursor = conn.cursor()
    final_insert_data = {}

    # Solo incluir columnas que existen en el metadata
    columnas_metadata = {meta['Variable'] for meta in metadata_list}

    for col_name in data_dict.keys():
        if col_name in columnas_metadata:
            final_insert_data[col_name] = data_dict.get(col_name)
```

```
    final_insert_data['estadovalidacion'] = estado_validacion

    for key, value in final_insert_data.items():
        if pd.isna(value):
            final_insert_data[key] = None

    sql_columns = ", ".join(final_insert_data.keys())
    sql_placeholders = ", ".join(["%s"] * len(final_insert_data))
    sql_insert = f"INSERT INTO declaraciones_pm_anuales ({sql_columns})
VALUES ({sql_placeholders})"

    valores = tuple(final_insert_data.values())

    try:
        cursor.execute(sql_insert, valores)
        conn.commit()
        id_declaracion = cursor.lastrowid
        return id_declaracion, None
    except mysql.connector.Error as err:
        rfc_val = data_dict.get('rfc_anon', 'Desconocido')
        ejercicio_val = data_dict.get('ejercicio', 'Desconocido')
        if err.errno == errorcode.ER_DUP_ENTRY:
             print(f"\nAdvertencia: Registro duplicado para RFC
{rfc_val} Ejercicio {ejercicio_val}. No se insertó.")
        else:
            print(f"\nError al insertar declaración para RFC {rfc_val}
Ejercicio {ejercicio_val}: {err}")
        conn.rollback()
        return None, err.errno
    finally:
        cursor.close()

# --- NUEVA FUNCIÓN DE LIMPIEZA ---
def limpiar_columnas_inutiles(df, chunk_num):
    """
    Identifica y elimina columnas que son 100% nulas o que solo
contienen ceros.
    """
    columnas_a_eliminar = []

    # Criterio 1: Columnas 100% vacías (NaN)
    cols_vacias = df.columns[df.isnull().all()].tolist()
    if cols_vacias:
```

```
        columnas_a_eliminar.extend(cols_vacias)

    # Criterio 2: Columnas que solo contienen ceros (y posiblemente
NaNs)
    # No queremos eliminar columnas clave como 'rfc_anon' o 'ejercicio'
    columnas_clave = ['rfc_anon', 'ejercicio']
    for col in df.columns:
        if col not in columnas_a_eliminar and col not in
columnas_clave:
            # Rellenamos los NaN con 0 para la comprobación y vemos si
todos los valores son 0
            if pd.to_numeric(df[col],
errors='coerce').fillna(0).eq(0).all():
                columnas_a_eliminar.append(col)

    # Eliminar duplicados de la lista y luego las columnas del
DataFrame
    columnas_a_eliminar = sorted(list(set(columnas_a_eliminar)))

    if columnas_a_eliminar:
        print(f"\n[Chunk {chunk_num}] Limpieza: Eliminando
{len(columnas_a_eliminar)} columnas inútiles (vacías o solo ceros):
{columnas_a_eliminar}")
        df_limpio = df.drop(columns=columnas_a_eliminar)
        return df_limpio
    else:
        print(f"\n[Chunk {chunk_num}] Limpieza: No se encontraron
columnas inútiles para eliminar.")
        return df

def etl_proceso_declaraciones(path_del_csv_actual, metadata_actual):
    conn = get_db_connection()
    if not conn:
        print(f"Proceso ETL para {path_del_csv_actual} abortado: fallo
de conexión a la BD.")
        return

    log_id = iniciar_log_procesamiento(conn, path_del_csv_actual)
    if not log_id:
        print(f"Proceso ETL para {path_del_csv_actual} abortado: fallo
al iniciar log.")
        if conn: conn.close()
        return
```

```
    df_iterator = read_csv_in_chunks(path_del_csv_actual, CHUNK_SIZE)

    if df_iterator is None:
        finalizar_log_procesamiento(conn, log_id, 'Fallido', 0,0,0,0,
f"Error crítico al leer/preprocesar {path_del_csv_actual}.")
        if conn: conn.close()
        return

    registros_procesados_global = 0
    registros_validos_cargados_global = 0
    registros_invalidos_global = 0
    total_registros_archivo_acumulado = 0

    print(f"\nIniciando procesamiento de chunks del archivo
{path_del_csv_actual}...")

    for i, chunk_df_original in enumerate(df_iterator):
        chunk_num = i + 1
        print(f"\n--- Procesando chunk {chunk_num} de
{path_del_csv_actual} ---")

        chunk_df = chunk_df_original.copy()
        chunk_df.columns = [col.lower() for col in chunk_df.columns] #
Asegurar minúsculas

        if chunk_df.empty:
            print("Chunk vacío, omitiendo.")
            continue

        # --- MODIFICACIÓN: LLAMAR A LA FUNCIÓN DE LIMPIEZA AQUÍ ---
        chunk_df = limpiar_columnas_inutiles(chunk_df, chunk_num)

        chunk_df = preprocess_chunk(chunk_df, metadata_actual)

        total_registros_chunk = len(chunk_df)
        total_registros_archivo_acumulado += total_registros_chunk
        print(f"Chunk {chunk_num} tiene {total_registros_chunk} filas
para procesar después de la limpieza.")

        for index, row in chunk_df.iterrows():
            registros_procesados_global += 1
            if registros_procesados_global % (max(1, CHUNK_SIZE // 20))
== 0 or total_registros_chunk < (max(1,CHUNK_SIZE // 20)) :
```

```
                print(f"\rProcesando fila global
{registros_procesados_global} (de archivo
{os.path.basename(path_del_csv_actual)})...", end="")

            fila_dict = row.to_dict()
            errores = validar_fila(row, metadata_actual, log_id, conn)

            id_declaracion_insertada = None
            insert_errno = None

            if not errores:
                id_declaracion_insertada, insert_errno =
insertar_declaracion(conn, fila_dict, 'Valido', metadata_actual)
                if id_declaracion_insertada:
                    registros_validos_cargados_global += 1
                else:
                    registros_invalidos_global += 1
                    if conn and insert_errno and insert_errno !=
errorcode.ER_DUP_ENTRY:
                        registrar_error_validacion(conn, log_id,
fila_dict.get('rfc_anon'), fila_dict.get('ejercicio'),
                                                   "GENERAL_INSERT",
str(fila_dict), "ERROR_INSERCION_BD", f"Fallo al insertar registro
validado (errno: {insert_errno}).")
            else:
                registros_invalidos_global += 1
                id_declaracion_insertada, insert_errno_invalid =
insertar_declaracion(conn, fila_dict, 'Invalido', metadata_actual)
                if id_declaracion_insertada and conn:
                    cursor_update = conn.cursor()
                    try:
                        rfc_ref_update = fila_dict.get('rfc_anon')
                        ejercicio_ref_update =
fila_dict.get('ejercicio')

                        if pd.notna(rfc_ref_update) and
pd.notna(ejercicio_ref_update): # Comprobar NaNs
                            sql_update_errores = """
                            UPDATE erroresvalidaciondeclaraciones
                            SET id_declaracion_fk = %s
                            WHERE logid_fk = %s AND rfc_anon_ref = %s
AND ejercicio_ref = %s AND id_declaracion_fk IS NULL
                            """
                            # Asegurarse que el ejercicio_ref sea int
```

```
                            cursor_update.execute(sql_update_errores,
(id_declaracion_insertada, log_id, rfc_ref_update,
int(ejercicio_ref_update)))
                            conn.commit()
                        except mysql.connector.Error as err_update:
                            print(f"\nError al actualizar FK en
ErroresValidacionDeclaraciones para ID {id_declaracion_insertada}:
{err_update}")
                            conn.rollback()
                        finally:
                            cursor_update.close()
                elif conn and insert_errno_invalid and
insert_errno_invalid != errorcode.ER_DUP_ENTRY:
                    registrar_error_validacion(conn, log_id,
fila_dict.get('rfc_anon'), fila_dict.get('ejercicio'),

"GENERAL_INSERT_INVALID", str(fila_dict),
"ERROR_INSERCION_BD_INVALIDO", f"Fallo al insertar registro inválido
(errno: {insert_errno_invalid})."

        print(f"\nFin del procesamiento del chunk {chunk_num} de
{path_del_csv_actual}.")

    print(f"\nProcesamiento de todos los chunks para
{path_del_csv_actual} completado. Total filas leídas:
{total_registros_archivo_acumulado}")

    mensaje_final = (f"Archivo:
{os.path.basename(path_del_csv_actual)}. Total Filas:
{total_registros_archivo_acumulado}, "
                     f"Procesadas: {registros_procesados_global},
Válidas Cargadas: {registros_validos_cargados_global}, "
                     f"Inválidas/No cargadas:
{registros_invalidos_global}.")

    estado_final = 'Completado'
    if registros_invalidos_global > 0:
        estado_final = 'CompletadoConErrores'
    if registros_procesados_global == 0 and
total_registros_archivo_acumulado > 0:
        estado_final = 'Fallido'

    finalizar_log_procesamiento(conn, log_id, estado_final,
```

```
                                total_registros_archivo_acumulado,
registros_procesados_global,
                                registros_validos_cargados_global,
registros_invalidos_global,
                                mensaje_final)

    if conn:
        conn.close()
        print(f"Conexión a la base de datos cerrada para el archivo
{os.path.basename(path_del_csv_actual)}.")

if __name__ == "__main__":
    print(f"--- INICIO DEL PROCESO ETL PARA {CSV_FILE_TO_PROCESS}
(Ejercicio: {EJERCICIO_FISCAL_ESPERADO}) ---")
    etl_proceso_declaraciones(CSV_FILE_TO_PROCESS, column_metadata) #
Pasar column_metadata
    print(f"--- FIN DEL PROCESO ETL PARA {CSV_FILE_TO_PROCESS} ---")
```

Anexo B: Código fuente de ETL_DWH.py.

```
import pandas as pd
import numpy as np
import mysql.connector
import time

# --- Configuración de la Base de Datos (la misma que antes) ---
DB_CONFIG = {
    'host': '127.0.0.1',
    'port': 3306,
    'user': 'root',
    'password': 'masterf01',
    'database': 'DeclaracionesAnualesDB'
}

BATCH_INSERT_SIZE = 10000
# *** CAMBIO IMPORTANTE AQUÍ: Nombre de la tabla OLTP generalizada ***
NOMBRE_TABLA_OLTP = 'declaraciones_pm_anuales'

def get_db_connection():
    try:
        conn = mysql.connector.connect(**DB_CONFIG)
        print("¡Conexión exitosa a la base de datos MySQL!")
        return conn
    except Exception as e:
        print(f"Error al conectar a la base de datos: {e}")
```

```
        return None

def limpiar_tablas_dw(conn):
    cursor = conn.cursor()
    print("Limpiando tablas del Data Warehouse...")
    try:
        cursor.execute("SET FOREIGN_KEY_CHECKS = 0;")
        cursor.execute("TRUNCATE TABLE fact_declaraciones;")
        cursor.execute("TRUNCATE TABLE dim_contribuyente;")
        cursor.execute("TRUNCATE TABLE dim_tiempo;")
        cursor.execute("TRUNCATE TABLE dim_calidad_dato;")
        cursor.execute("SET FOREIGN_KEY_CHECKS = 1;")
        conn.commit()
        print("Tablas del DW limpiadas exitosamente.")
    except Exception as e:
        print(f"Error al limpiar las tablas del DW: {e}")
        conn.rollback()
    finally:
        cursor.close()

def poblar_dim_tiempo(conn):
    cursor = conn.cursor()
    print("Poblando dim_tiempo...")
    try:
        # *** CAMBIO IMPORTANTE AQUÍ: Usar el nombre de tabla correcto
***
        sql_select = f"SELECT DISTINCT ejercicio FROM
{NOMBRE_TABLA_OLTP};"
        cursor.execute(sql_select)
        ejercicios = cursor.fetchall()

        sql_insert = "INSERT IGNORE INTO dim_tiempo (ejercicio) VALUES
(%s);"
        cursor.executemany(sql_insert, ejercicios)
        conn.commit()
        print(f"dim_tiempo poblada con {cursor.rowcount} nuevo(s)
registro(s).")
    except Exception as e:
        print(f"Error al poblar dim_tiempo: {e}")
        conn.rollback()
    finally:
        cursor.close()

def poblar_dim_contribuyente(conn):
```

```
    cursor = conn.cursor()
    print("Poblando dim_contribuyente...")
    try:
        # *** CAMBIO IMPORTANTE AQUÍ: Usar el nombre de tabla correcto
***
        sql_insert_select = f"""
        INSERT INTO dim_contribuyente (rfc_anon)
        SELECT DISTINCT rfc_anon FROM {NOMBRE_TABLA_OLTP};
        """
        cursor.execute(sql_insert_select)
        conn.commit()
        print(f"dim_contribuyente poblada con {cursor.rowcount}
registros.")
    except Exception as e:
        print(f"Error al poblar dim_contribuyente: {e}")
        conn.rollback()
    finally:
        cursor.close()

def poblar_dim_calidad_dato(conn):
    cursor = conn.cursor()
    print("Poblando dim_calidad_dato...")
    try:
        categorias = [
            ('Valido', 'NINGUNO'),
            ('Invalido', 'INCONSISTENCIA_UPAPTU'),
            ('Invalido', 'INCONSISTENCIA_TOTES'),
            ('Invalido', 'OTRO')
        ]
        sql_insert = "INSERT INTO dim_calidad_dato (estado_validacion,
codigo_error_principal) VALUES (%s, %s);"
        cursor.executemany(sql_insert, categorias)
        conn.commit()
        print(f"dim_calidad_dato poblada con {cursor.rowcount}
categorías.")
    except Exception as e:
        print(f"Error al poblar dim_calidad_dato: {e}")
        conn.rollback()
    finally:
        cursor.close()

def poblar_fact_declaraciones(conn):
    print("\nIniciando población de fact_declaraciones...")
```

```
    print("Cargando dimensiones en memoria...")
    tiempo_map = pd.read_sql("SELECT tiempo_key, ejercicio FROM
dim_tiempo", conn).set_index('ejercicio')['tiempo_key']
    contrib_map = pd.read_sql("SELECT contribuyente_key, rfc_anon FROM
dim_contribuyente", conn).set_index('rfc_anon')['contribuyente_key']
    calidad_df = pd.read_sql("SELECT calidad_key, estado_validacion,
codigo_error_principal FROM dim_calidad_dato", conn)
    calidad_df['map_key'] = list(zip(calidad_df['estado_validacion'],
calidad_df['codigo_error_principal']))
    calidad_map = calidad_df.set_index('map_key')['calidad_key']
    print("Dimensiones cargadas.")

    # *** CAMBIO IMPORTANTE AQUÍ: Usar el nombre de tabla correcto ***
    sql_extract = f"""
    SELECT
        d.*,
        (SELECT e.codigoerror FROM erroresvalidaciondeclaraciones e
         WHERE e.id_declaracion_fk = d.id_declaracion
         ORDER BY e.errorid LIMIT 1) as codigo_error
    FROM
        {NOMBRE_TABLA_OLTP} d
    """

    metric_columns = [
        'it_aa', 'td_aa', 'upaptu_c_aa', 'ptu_aa', 'ufe_c_aa',
'pfe_c_aa',
        'pfea_aa', 'dafpe_aa', 'rfis_c_aa', 'isreje_c_aa', 'cfdim_aa',
'rpm_aa',
        'orisr_aa', 'impceje_c_aa', 'estec_aa', 'escine_aa',
'estea_aa',
        'otroes_aa', 'totes_c_aa', 'ppcon_aa', 'ppef_aa', 'imrc_aa',
'imape_aa',
        'imad_aa', 'cfietu_aa', 'imccfc_aa', 'imccfe_aa', 'occ_aa',
'ocf_aa',
        'difcar_c_aa', 'diffav_c_aa', 'imide_aa', 'impsun_aa',
'isrcareje_c_aa',
        'isrpeimpac_aa', 'isrfaveje_c_aa'
    ]

    total_rows_inserted_in_fact = 0
    start_time = time.time()

    print(f"Leyendo todos los datos de {NOMBRE_TABLA_OLTP} para
fact_declaraciones...")
```

```
    try:
        df_oltp_completo_original = pd.read_sql(sql_extract, conn)
        print(f"Se leyeron {len(df_oltp_completo_original)} filas de la
tabla OLTP.")
    except Exception as e_read:
        print(f"Error al leer todos los datos de OLTP: {e_read}")
        return

    if not df_oltp_completo_original.empty:
        df_processed = df_oltp_completo_original.copy()
        df_processed.columns = [col.lower() for col in
df_processed.columns]

        if 'rfc_anon' not in df_processed.columns:
            print("ERROR CRÍTICO: La columna 'rfc_anon' NO está en el
DataFrame df_processed DESPUÉS de forzar minúsculas.")
            return

        print(f"Procesando el DataFrame completo para la tabla de
hechos...")

        df_processed['contribuyente_key'] =
df_processed['rfc_anon'].map(contrib_map)
        df_processed['tiempo_key'] =
df_processed['ejercicio'].map(tiempo_map)

        df_processed['codigo_error_principal'] =
df_processed['codigo_error'].fillna(
            df_processed['estadovalidacion'].apply(lambda x: 'NINGUNO'
if x == 'Valido' else 'OTRO')
        )
        df_processed['calidad_map_key'] =
list(zip(df_processed['estadovalidacion'],
df_processed['codigo_error_principal']))
        df_processed['calidad_key'] =
df_processed['calidad_map_key'].map(calidad_map)

        if df_processed['contribuyente_key'].isnull().any():
            print(f"ADVERTENCIA: Se encontraron
{df_processed['contribuyente_key'].isnull().sum()} valores nulos en
'contribuyente_key'.")
        if df_processed['tiempo_key'].isnull().any():
```

```
            print(f"ADVERTENCIA: Se encontraron
{df_processed['tiempo_key'].isnull().sum()} valores nulos en
'tiempo_key'.")
        if df_processed['calidad_key'].isnull().any():
            print(f"ADVERTENCIA: Se encontraron
{df_processed['calidad_key'].isnull().sum()} valores nulos en
'calidad_key'.")

        original_rows = len(df_processed)
        df_processed.dropna(subset=['contribuyente_key', 'tiempo_key',
'calidad_key'], inplace=True)
        if len(df_processed) < original_rows:
            print(f"ADVERTENCIA: Se descartaron {original_rows -
len(df_processed)} filas debido a claves de dimensión faltantes.")

        if not df_processed.empty:
            columnas_para_df_for_insert = ['contribuyente_key',
'tiempo_key', 'calidad_key'] + metric_columns

            columnas_faltantes = [col for col in
columnas_para_df_for_insert if col not in df_processed.columns]
            if columnas_faltantes:
                print(f"ERROR CRÍTICO: Las siguientes columnas NO
existen en df_processed: {columnas_faltantes}")
                return

            df_for_insert =
df_processed[columnas_para_df_for_insert].copy()

            print("\n--- INFO: Preparando tuplas y convirtiendo NaN a
None ---")
            temp_records = df_for_insert.to_dict(orient='records')
            list_of_tuples_to_insert = []
            for record in temp_records:
                list_of_tuples_to_insert.append(
                    tuple(None if pd.isna(record[col]) else record[col]
for col in df_for_insert.columns)
                )

            cursor = conn.cursor(buffered=True)

            lista_columnas_sql = df_for_insert.columns.tolist()
            columnas_sql_string = ', '.join(lista_columnas_sql)
```

```
            placeholders_sql_string = ', '.join(['%s'] *
len(lista_columnas_sql))

            sql_insert_fact = f"""
            INSERT INTO fact_declaraciones
            ({columnas_sql_string})
            VALUES ({placeholders_sql_string})
            """

            total_rows_to_insert = len(list_of_tuples_to_insert)

            for i_batch in range(0, total_rows_to_insert,
BATCH_INSERT_SIZE):
                batch_to_insert = list_of_tuples_to_insert[i_batch :
i_batch + BATCH_INSERT_SIZE]
                if not batch_to_insert:
                    continue

                current_batch_size = len(batch_to_insert)
                print(f"Intentando insertar lote de
{current_batch_size} filas (total insertadas hasta ahora:
{total_rows_inserted_in_fact})...")
                try:
                    cursor.executemany(sql_insert_fact,
batch_to_insert)
                    conn.commit()
                    total_rows_inserted_in_fact += current_batch_size
                    print(f"Lote insertado.
{total_rows_inserted_in_fact} de {total_rows_to_insert} filas
insertadas en total en fact_declaraciones.")
                except mysql.connector.Error as err_insert:
                    print(f"Error durante executemany en
fact_declaraciones (lote iniciando en índice {i_batch}): {err_insert}")
                    conn.rollback()
                    print("Deteniendo la inserción de lotes debido a un
error.")
                    break
            cursor.close()
        else:
            print(f"DataFrame vacío después de filtrar claves de
dimensión nulas. No se insertaron datos en fact_declaraciones.")
    else:
        print(f"No se leyeron datos de la tabla OLTP
({NOMBRE_TABLA_OLTP}) para poblar fact_declaraciones.")
```

```
    end_time = time.time()
    print(f"\nProceso de población de fact_declaraciones finalizado en
{end_time - start_time:.2f} segundos.")

def run_dw_etl():
    conn = get_db_connection()
    if not conn:
        print("Proceso ETL abortado.")
        return

    try:
        limpiar_tablas_dw(conn)
        poblar_dim_tiempo(conn)
        poblar_dim_contribuyente(conn)
        poblar_dim_calidad_dato(conn)
        poblar_fact_declaraciones(conn)

        print("\n¡Proceso ETL de OLTP a DW finalizado!")

    except Exception as e:
        print(f"Ocurrió un error general en el proceso ETL del DW:
{e}")
    finally:
        if conn:
            conn.close()
            print("Conexión a la base de datos cerrada.")

if __name__ == "__main__":
    run_dw_etl()
```

Anexo C: Código fuente de OLTP.sql.

```
-- ============================================
-- FASE 0: CREACIÓN DE LA BASE DE DATOS
-- ============================================
CREATE DATABASE IF NOT EXISTS DeclaracionesAnualesDB;
USE DeclaracionesAnualesDB;

-- ============================================
-- FASE 1: ESTRUCTURA OLTP
-- (Tabla para almacenar datos de CSV, logs y errores)
-- ============================================
```

```sql
-- Tabla principal OLTP para las Declaraciones Anuales de Personas
Morales
CREATE TABLE IF NOT EXISTS declaraciones_pm_anuales (
    id_declaracion INT AUTO_INCREMENT PRIMARY KEY,
    rfc_anon BIGINT NOT NULL,
    ejercicio INT NOT NULL,
    it_aa DECIMAL(20, 4) NULL,
    td_aa DECIMAL(20, 4) NULL,
    upaptu_c_aa DECIMAL(20, 4) NULL,
    ptu_aa DECIMAL(20, 4) NULL,
    ufe_c_aa DECIMAL(20, 4) NULL,
    pfe_c_aa DECIMAL(20, 4) NULL,
    pfea_aa DECIMAL(20, 4) NULL,
    dafpe_aa DECIMAL(20, 4) NULL,
    rfis_c_aa DECIMAL(20, 4) NULL,
    isreje_c_aa DECIMAL(20, 4) NULL,
    cfdim_aa DECIMAL(20, 4) NULL,
    rpm_aa DECIMAL(20, 4) NULL,
    orisr_aa DECIMAL(20, 4) NULL,
    impceje_c_aa DECIMAL(20, 4) NULL,
    estec_aa DECIMAL(20, 4) NULL,
    escine_aa DECIMAL(20, 4) NULL,
    estea_aa DECIMAL(20, 4) NULL,
    otroes_aa DECIMAL(20, 4) NULL,
    totes_c_aa DECIMAL(20, 4) NULL,
    ppcon_aa DECIMAL(20, 4) NULL,
    ppef_aa DECIMAL(20, 4) NULL,
    imrc_aa DECIMAL(20, 4) NULL,
    imape_aa DECIMAL(20, 4) NULL,
    imad_aa DECIMAL(20, 4) NULL,
    cfietu_aa DECIMAL(20, 4) NULL,
    imccfc_aa DECIMAL(20, 4) NULL,
    imccfe_aa DECIMAL(20, 4) NULL,
    occ_aa DECIMAL(20, 4) NULL,
    ocf_aa DECIMAL(20, 4) NULL,
    difcar_c_aa DECIMAL(20, 4) NULL,
    diffav_c_aa DECIMAL(20, 4) NULL,
    imide_aa DECIMAL(20, 4) NULL,
    impsun_aa DECIMAL(20, 4) NULL,
    isrcareje_c_aa DECIMAL(20, 4) NULL,
    isrpeimpac_aa DECIMAL(20, 4) NULL,
    isrfaveje_c_aa DECIMAL(20, 4) NULL,
    fechacarga DATETIME DEFAULT CURRENT_TIMESTAMP,
    estadovalidacion VARCHAR(50) DEFAULT 'Pendiente',
```

```
    CONSTRAINT uq_rfc_ejercicio UNIQUE (rfc_anon, ejercicio)
);

-- Tabla para el log de procesamiento de archivos (ETL CSV -> OLTP)
CREATE TABLE IF NOT EXISTS procesamientologdeclaraciones (
    logid INT AUTO_INCREMENT PRIMARY KEY,
    nombrearchivo VARCHAR(255),
    fechahorainicio DATETIME,
    fechahorafin DATETIME NULL,
    estadogeneral VARCHAR(50),
    totalregistrosarchivo INT NULL,
    registrosprocesados INT DEFAULT 0,
    registrosvalidoscargados INT DEFAULT 0,
    registrosinvalidos INT DEFAULT 0,
    mensaje TEXT NULL,
    usuarioejecucion VARCHAR(100) NULL
);

-- Tabla para registrar errores de validación (ETL CSV -> OLTP)
CREATE TABLE IF NOT EXISTS erroresvalidaciondeclaraciones (
    errorid INT AUTO_INCREMENT PRIMARY KEY,
    id_declaracion_fk INT NULL,
    rfc_anon_ref BIGINT NULL,
    ejercicio_ref INT NULL,
    logid_fk INT,
    nombrecampo VARCHAR(100),
    valorcampoproblematico TEXT,
    codigoerror VARCHAR(50),
    descripcionerror TEXT,
    fechahoraerror DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (id_declaracion_fk) REFERENCES
declaraciones_pm_anuales(id_declaracion) ON DELETE CASCADE,
    FOREIGN KEY (logid_fk) REFERENCES
procesamientologdeclaraciones(logid) ON DELETE CASCADE
);

-- ============================================
-- FASE 2: ESTRUCTURA DEL DATA WAREHOUSE (Para ETL OLTP -> DW)
-- ============================================

-- DIMENSIÓN 1: Contribuyente
CREATE TABLE IF NOT EXISTS dim_contribuyente (
    contribuyente_key INT AUTO_INCREMENT PRIMARY KEY,
    rfc_anon BIGINT NOT NULL,
```

```
    CONSTRAINT uq_dim_rfc_anon UNIQUE (rfc_anon)
);

-- DIMENSIÓN 2: Tiempo
CREATE TABLE IF NOT EXISTS dim_tiempo (
    tiempo_key INT AUTO_INCREMENT PRIMARY KEY,
    ejercicio INT NOT NULL,
    CONSTRAINT uq_dim_ejercicio UNIQUE (ejercicio)
);

-- DIMENSIÓN 3: Calidad del Dato
CREATE TABLE IF NOT EXISTS dim_calidad_dato (
    calidad_key INT AUTO_INCREMENT PRIMARY KEY,
    estado_validacion VARCHAR(50) NOT NULL,
    codigo_error_principal VARCHAR(50) NOT NULL,
    CONSTRAINT uq_dim_calidad UNIQUE (estado_validacion,
codigo_error_principal)
);

-- TABLA DE HECHOS: Declaraciones Anuales
CREATE TABLE IF NOT EXISTS fact_declaraciones (
    declaracion_key BIGINT AUTO_INCREMENT PRIMARY KEY,
    contribuyente_key INT NOT NULL,
    tiempo_key INT NOT NULL,
    calidad_key INT NOT NULL,
    it_aa DECIMAL(20, 4) NULL,
    td_aa DECIMAL(20, 4) NULL,
    upaptu_c_aa DECIMAL(20, 4) NULL,
    ptu_aa DECIMAL(20, 4) NULL,
    ufe_c_aa DECIMAL(20, 4) NULL,
    pfe_c_aa DECIMAL(20, 4) NULL,
    pfea_aa DECIMAL(20, 4) NULL,
    dafpe_aa DECIMAL(20, 4) NULL,
    rfis_c_aa DECIMAL(20, 4) NULL,
    isreje_c_aa DECIMAL(20, 4) NULL,
    cfdim_aa DECIMAL(20, 4) NULL,
    rpm_aa DECIMAL(20, 4) NULL,
    orisr_aa DECIMAL(20, 4) NULL,
    impceje_c_aa DECIMAL(20, 4) NULL,
    estec_aa DECIMAL(20, 4) NULL,
    escine_aa DECIMAL(20, 4) NULL,
    estea_aa DECIMAL(20, 4) NULL,
    otroes_aa DECIMAL(20, 4) NULL,
    totes_c_aa DECIMAL(20, 4) NULL,
```

```
    ppcon_aa DECIMAL(20, 4) NULL,
    ppef_aa DECIMAL(20, 4) NULL,
    imrc_aa DECIMAL(20, 4) NULL,
    imape_aa DECIMAL(20, 4) NULL,
    imad_aa DECIMAL(20, 4) NULL,
    cfietu_aa DECIMAL(20, 4) NULL,
    imccfc_aa DECIMAL(20, 4) NULL,
    imccfe_aa DECIMAL(20, 4) NULL,
    occ_aa DECIMAL(20, 4) NULL,
    ocf_aa DECIMAL(20, 4) NULL,
    difcar_c_aa DECIMAL(20, 4) NULL,
    diffav_c_aa DECIMAL(20, 4) NULL,
    imide_aa DECIMAL(20, 4) NULL,
    impsun_aa DECIMAL(20, 4) NULL,
    isrcareje_c_aa DECIMAL(20, 4) NULL,
    isrpeimpac_aa DECIMAL(20, 4) NULL,
    isrfaveje_c_aa DECIMAL(20, 4) NULL,
    FOREIGN KEY (contribuyente_key) REFERENCES
dim_contribuyente(contribuyente_key),
    FOREIGN KEY (tiempo_key) REFERENCES dim_tiempo(tiempo_key),
    FOREIGN KEY (calidad_key) REFERENCES dim_calidad_dato(calidad_key)
);

-- ============================================
-- FASE 3: CONSULTAS DE VERIFICACIÓN Y ANÁLISIS
-- ============================================

-- ----- Consultas de Negocio para el OLTP (Punto 'c' del Proyecto) ---
--

-- 1. Resumen de la última ejecución del ETL
-- Proporciona una visión rápida del resultado del proceso de carga más
reciente, útil para verificar el estado.
SELECT * FROM procesamientologdeclaraciones ORDER BY logid DESC LIMIT
2;

-- 2. Conteo de registros por estado de validación para una carga
específica
-- Permite auditar la calidad de los datos de un archivo en particular
(reemplazar [ID_LOG] con un ID real).

SELECT
    d.estadovalidacion,
    COUNT(d.id_declaracion) AS total_declaraciones
```

```sql
FROM declaraciones_pm_anuales d
WHERE EXISTS (
    SELECT 1 FROM erroresvalidaciondeclaraciones e
    WHERE e.id_declaracion_fk = d.id_declaracion AND e.logid_fk = 2
)
OR d.estadovalidacion = 'Valido'
GROUP BY d.estadovalidacion;

-- 3. Frecuencia de los tipos de error en la última carga
-- Ayuda a identificar los problemas de calidad de datos más comunes en la fuente de datos más reciente.
SELECT codigoerror, COUNT(*) AS frecuencia
FROM erroresvalidaciondeclaraciones
WHERE logid_fk = (SELECT MAX(logid) FROM procesamientologdeclaraciones)
GROUP BY codigoerror
ORDER BY frecuencia DESC;

-- 4. Muestra de registros con un error específico
-- Facilita el análisis a fondo de un problema particular, mostrando ejemplos concretos.
SELECT * FROM declaraciones_pm_anuales
WHERE id_declaracion IN (
    SELECT id_declaracion_fk FROM erroresvalidaciondeclaraciones
    WHERE codigoerror = 'INCONSISTENCIA_UPAPTU' AND logid_fk = (SELECT MAX(logid) FROM procesamientologdeclaraciones)
) LIMIT 10;

-- 5. Verificación de duplicados cargados en la tabla OLTP
-- Es una consulta operativa clave para garantizar la integridad de los datos y que la constraint UNIQUE está funcionando.
SELECT rfc_anon, ejercicio, COUNT(*) AS numero_de_registros
FROM declaraciones_pm_anuales
GROUP BY rfc_anon, ejercicio
HAVING COUNT(*) > 1;


-- ----- Consultas de Verificación del DW (Después de ETL_DWH.py) -----
-- Verificar contenido de dim_tiempo
SELECT * FROM dim_tiempo ORDER BY ejercicio;

-- Contar el total de hechos en fact_declaraciones
SELECT COUNT(*) AS total_hechos FROM fact_declaraciones;

-- Muestra de datos de la tabla de hechos con dimensiones
```

```sql
SELECT
    fd.declaracion_key,
    dc.rfc_anon,
    dt.ejercicio,
    dqd.estado_validacion,
    dqd.codigo_error_principal,
    fd.it_aa,
    fd.td_aa,
    fd.upaptu_c_aa
FROM fact_declaraciones fd
JOIN dim_contribuyente dc ON fd.contribuyente_key =
dc.contribuyente_key
JOIN dim_tiempo dt ON fd.tiempo_key = dt.tiempo_key
JOIN dim_calidad_dato dqd ON fd.calidad_key = dqd.calidad_key
ORDER BY dt.ejercicio, fd.declaracion_key
LIMIT 20;


-- ----- Consultas de Negocio para el DW (Punto 'g' del Proyecto) -----

-- 1. Ingresos Acumulables Totales (IT_AA) y Resultado Fiscal Total
(RFIS_c_AA) por Año:
SELECT
    t.ejercicio,
    SUM(f.it_aa) AS TotalIngresosAcumulables,
    SUM(f.rfis_c_aa) AS TotalResultadoFiscal
FROM fact_declaraciones f
JOIN dim_tiempo t ON f.tiempo_key = t.tiempo_key
GROUP BY t.ejercicio
ORDER BY t.ejercicio;

-- 2. Distribución de Declaraciones por Estado de Validación y Código
de Error Principal, por Año:
SELECT
    t.ejercicio,
    qd.estado_validacion,
    qd.codigo_error_principal,
    COUNT(f.declaracion_key) AS NumeroDeDeclaraciones
FROM fact_declaraciones f
JOIN dim_tiempo t ON f.tiempo_key = t.tiempo_key
JOIN dim_calidad_dato qd ON f.calidad_key = qd.calidad_key
GROUP BY t.ejercicio, qd.estado_validacion, qd.codigo_error_principal
ORDER BY t.ejercicio, NumeroDeDeclaraciones DESC;
```

```sql
-- 3. Comparativa Anual del ISR del Ejercicio (ISREJE_c_AA) para
Declaraciones "Válidas" vs. "INCONSISTENCIA_UPAPTU":
SELECT
    t.ejercicio,
    qd.codigo_error_principal,
    AVG(f.isreje_c_aa) AS Promedio_ISR_Ejercicio,
    COUNT(f.declaracion_key) AS NumeroDeDeclaraciones
FROM fact_declaraciones f
JOIN dim_tiempo t ON f.tiempo_key = t.tiempo_key
JOIN dim_calidad_dato qd ON f.calidad_key = qd.calidad_key
WHERE qd.codigo_error_principal IN ('NINGUNO', 'INCONSISTENCIA_UPAPTU')
GROUP BY t.ejercicio, qd.codigo_error_principal
ORDER BY t.ejercicio, qd.codigo_error_principal;

-- 4. Top 5 RFCs por Suma de Pérdidas Fiscales del Ejercicio (PFE_c_AA)
en 2015:
SELECT
    c.rfc_anon,
    SUM(f.pfe_c_aa) AS TotalPerdidaFiscal_2015
FROM fact_declaraciones f
JOIN dim_contribuyente c ON f.contribuyente_key = c.contribuyente_key
JOIN dim_tiempo t ON f.tiempo_key = t.tiempo_key
WHERE t.ejercicio = 2015 AND f.pfe_c_aa > 0
GROUP BY c.rfc_anon
ORDER BY TotalPerdidaFiscal_2015 DESC
LIMIT 5;

-- 5. Evolución del Promedio de PTU Pagada (PTU_AA) por Año (solo para
quienes pagaron PTU):
SELECT
    t.ejercicio,
    AVG(f.ptu_aa) AS Promedio_PTU_Pagada,
    COUNT(DISTINCT f.contribuyente_key) AS
NumeroDeEmpresasQuePagaronPTU
FROM fact_declaraciones f
JOIN dim_tiempo t ON f.tiempo_key = t.tiempo_key
WHERE f.ptu_aa > 0
GROUP BY t.ejercicio
ORDER BY t.ejercicio;
-- ----- Sentencias para Limpiar Datos (USAR CON PRECAUCIÓN) -----

SET SQL_SAFE_UPDATES = 0;

-- Limpiar DW
```

```sql
DELETE FROM fact_declaraciones;
DELETE FROM dim_calidad_dato;
DELETE FROM dim_tiempo;
DELETE FROM dim_contribuyente;

-- Limpiar Errores y Logs del OLTP
DELETE FROM erroresvalidaciondeclaraciones;
DELETE FROM procesamientologdeclaraciones;

-- Limpiar Tabla OLTP Principal
DELETE FROM declaraciones_pm_anuales;

SET SQL_SAFE_UPDATES = 1;

-- Para reiniciar los contadores AUTO_INCREMENT (opcional)
-- ALTER TABLE declaraciones_pm_anuales AUTO_INCREMENT = 1;
-- ALTER TABLE procesamientologdeclaraciones AUTO_INCREMENT = 1;
-- ALTER TABLE erroresvalidaciondeclaraciones AUTO_INCREMENT = 1;
-- ALTER TABLE dim_contribuyente AUTO_INCREMENT = 1;
-- ALTER TABLE dim_tiempo AUTO_INCREMENT = 1;
-- ALTER TABLE dim_calidad_dato AUTO_INCREMENT = 1;
-- ALTER TABLE fact_declaraciones AUTO_INCREMENT = 1;

SELECT 'Script SQL consolidado y estructurado listo para revisión y
uso.' AS Estado_Final_Script;
```