Documentación: Script de Carga de Órdenes Pendientes v2.0

Este documento detalla el funcionamiento, uso y mantenimiento del script ETL (Extracción, Transformación y Carga) diseñado para procesar y cargar datos de órdenes de venta pendientes en una base de datos SQL Server con enriquecimiento temporal automático y validación robusta de datos.

1. Funcionalidades Principales

El script automatiza el proceso de actualizar la base de datos con información de órdenes pendientes, implementando una estrategia de snapshot diario con transformaciones avanzadas de datos y enriquecimiento temporal.

Configuración y Conexión Segura

- **Variables de Entorno:** Utiliza un archivo `.env` para gestionar las credenciales de la base de datos de forma segura.
- **Compatibilidad PyInstaller:** Detecta automáticamente si se ejecuta como ejecutable (.exe) o en entorno de desarrollo mediante la función `get_env_path()`.
- **Conexión Robusta:** Establece conexión segura con SQL Server usando SQLAlchemy con validación previa (`SELECT 1`) antes de procesar datos.
- **Manejo de Recursos:** Disposición automática de conexiones en bloque `finally` para garantizar liberación de recursos.

Selección y Procesamiento de Archivos Especializado

Carga Inteligente de CSV:

- **Interfaz Específica:** Diálogo optimizado para archivos de órdenes pendientes
- **Omisión de Encabezados/Pie:** Utiliza `skiprows=6` y `skipfooter=1` para manejar formato específico del reporte
- **Motor Python:** Usa `engine='python'` para compatibilidad con `skipfooter`
- **Validación de Carga:** Confirmación exitosa con mensaje específico

Transformación y Estandarización de Datos

Mapeo de Columnas Dinámico:

Conversión inteligente de nombres de columnas del sistema origen:

- `Customer` → `nombre_cliente`

- `Amount (Net)` → `amount_net`
- `Document Number` → `document_number`
- `Date` → `fecha`
- `Class Item` → `class_item`
- `Quantity` → `cantidad`
- **Estado Flexible:** Maneja tanto `Validated Status` como `Status` → `estado`

Procesamiento de Datos Especializado:

- **Class Item Default:** Asigna "Descuento" a valores nulos en `class_item`
- **Limpieza de Tipos:** Eliminación de símbolos monetarios y separadores de miles
- **Validación Numérica:** Conversión segura con valores por defecto apropiados

Enriquecimiento Temporal Automático

Generación de Dimensiones Temporales:

A partir de la columna `fecha`, el script genera automáticamente:

- `nombre_mes`**:** Nombre completo del mes en formato texto
- `mes`**:** Número del mes (1-12)
- `dia`**:** Día del mes
- `año`**:** Año de la orden

Procesamiento de Fechas Robusto:

- **Normalización:** Convierte fechas a medianoche con `dt.normalize()`
- **Manejo de Errores:** Usa `errors='coerce'` para fechas inválidas
- **Valor por Defecto:** Asigna `1900-01-01` para fechas no convertibles
- **Formato Estándar:** Garantiza consistencia temporal

Enriquecimiento de Datos con Base de Datos

Mapeo Dinámico de Clientes y Zonas:

- **Consulta Completa:** Extrae `id_cliente`, `nombre_cliente` e `id_zone` de la tabla `Clientes`
- **Limpieza Bidireccional:** Aplica función `clean_customer_name()` a ambos datasets
- **Merge Inteligente:** Utiliza `pd.merge()` con `how='left'` para preservar órdenes
- **Zona por Defecto:** Asigna `DEFAULT_ZONE_ID = 1` cuando el cliente no tiene zona

Función de Limpieza Avanzada:

```python
def clean_customer_name(name):
    # Manejo de valores nulos
    # Normalización a minúsculas
    # Eliminación de caracteres especiales con regex
    # Normalización de espacios múltiples
```

Validación de Integridad:

- **Filtrado de No Mapeados:** Elimina registros sin `id_cliente` válido
- **Conversión Segura:** Asegura tipos enteros para IDs
- **Reporting Detallado:** Lista específica de clientes no encontrados

Transformación Final y Limpieza de Datos

Procesamiento de Campos Monetarios:

- **Amount Net:** Eliminación de `$`, `,` y espacios, conversión a float con default 0.0
- **Cantidad:** Eliminación de separadores de miles, conversión a entero

Validación de Campos de Texto:

- **Document Number:** Limitado a 20 caracteres, conversión a string
- **Estado:** Limitado a 50 caracteres, default "Desconocido"
- **Normalización:** Eliminación de espacios en extremos

Estructura Final:

Organización de columnas en formato específico para la tabla de destino:

```python
final_db_columns = [
    'id_cliente', 'class_item', 'cantidad', 'amount_net', 'document_number',
    'estado', 'fecha', 'id_zone', 'nombre_mes', 'mes', 'dia', 'año'
]
```

Estrategia de Snapshot Diario

Concepto de "Foto Diaria":

- **Carga Completa:** Todos los registros válidos del archivo se insertan
- **Marca Temporal:** Columna `FechaCarga` con `datetime.date.today()`
- **Sin Deduplicación:** Cada ejecución representa el estado completo del día
- **Análisis Temporal:** Permite comparaciones entre diferentes fechas de carga

2. Arquitectura de Datos

Tabla de Destino: `Pending_Orders`

Estructura esperada para recibir los datos procesados:

- `id_cliente` (INT, FK a tabla Clientes)
- `class_item` (VARCHAR, categoría del ítem)
- `cantidad` (INT, cantidad de productos)
- `amount_net` (DECIMAL/MONEY, monto neto)
- `document_number` (VARCHAR(20), número de documento)
- `estado` (VARCHAR(50), estado de la orden)
- `fecha` (DATE, fecha de la orden)
- `id_zone` (INT, zona del cliente)
- `nombre_mes` (VARCHAR, nombre del mes)
- `mes` (INT, número del mes)
- `dia` (INT, día del mes)
- `año` (INT, año)
- `FechaCarga` (DATE, timestamp de la carga)

Tabla de Referencia: `Clientes`

- `id_cliente` (INT, PK)
- `nombre_cliente` (VARCHAR)
- `id_zone` (INT, zona asignada al cliente)

Configuraciones de Base de Datos

```env
```

```
SERVER_NAME=servidor_ordenes
PORT=1433
DATABASE_NAME=VentasDB
DB_USERNAME=usuario_ordenes
DB_PASSWORD=password_seguro
```

3. Manejo y Uso del Script

Distribución

El script se distribuye como **archivo ejecutable (.exe)** optimizado para uso por personal de ventas y operaciones sin conocimientos técnicos.

Requisitos Previos

1. Archivo de Configuración (.env)

Debe ubicarse en la misma carpeta que el ejecutable:

```env
SERVER_NAME=servidor_ventas
PORT=1433
DATABASE_NAME=SistemaVentas
DB_USERNAME=usuario_ordenes
DB_PASSWORD=clave_acceso
```

2. Estructura de Base de Datos

- **Tabla `Clientes`:** Debe contener `id_cliente`, `nombre_cliente` e `id_zone`
- **Tabla `Pending_Orders`:** Estructura compatible con el esquema definido
- **Permisos:** Usuario debe tener permisos de lectura en `Clientes` y escritura en `Pending_Orders`

3. Archivo CSV de Órdenes

- **Formato:** Archivo CSV con estructura específica del sistema de órdenes
- **Encabezados:** 6 líneas iniciales que se omiten automáticamente
- **Pie de Página:** 1 línea final que se omite automáticamente
- **Columnas:** Estructura específica con nombres en inglés

Pasos para Ejecución

1. Preparación del Archivo

- Exportar reporte de órdenes pendientes desde el sistema de ventas
- Guardar como archivo CSV con nombre descriptivo (ej: `ordenes_pendientes_2024-01-15.csv`)

2. Ejecución del Script

- Ejecutar el archivo `.exe`
- El sistema valida automáticamente la conexión:

```
Conexión a SQL Server 'SistemaVentas' en 'servidor_ventas' establecida.
```

3. Selección de Archivo

- Se presenta diálogo específico para archivos CSV:

```
Por favor, selecciona el archivo 'ordenes_pendientes.csv'...
```

4. Procesamiento Automático

La consola muestra el progreso detallado:

```
CSV cargado exitosamente.
Columnas renombradas.
Columnas de fecha procesadas.
Mapeando clientes y zonas desde la tabla Clientes...
Mapeo de clientes y zonas finalizado.
Realizando limpieza final...
Limpieza final completada.
```

5. Carga Completa

```
Total de filas en el DataFrame preparado: 2500
Filas a insertar (snapshot diario completo): 2500
Iniciando inserción por lotes en la tabla 'Pending_Orders'...
Proceso de carga finalizado. Total de filas insertadas: 2500
Recursos de la base de datos liberados.
```

4. Mejoras de la Versión 2.0

Diferencias vs Versión 1.1

Aspecto	Versión 1.1	Versión 2.0
Enriquecimiento Temporal	Básico	Generación automática de dimensiones temporales
Mapeo de Zonas	Separado	Integrado con mapeo de clientes
Limpieza de Datos	Función externa	Función `clean_customer_name()` integrada
Manejo de Estados	Fijo	Flexible (Validated/Status)
Gestión de Recursos	Básica	Bloque `finally` para liberación garantizada
Validación de BD	Manual	Verificación automática de columna `id_zone`

Nuevas Funcionalidades v2.0

Enriquecimiento Temporal Automático:

- **Dimensiones Temporales:** Generación automática de mes, día, año, nombre_mes
- **Análisis Facilitado:** Campos listos para reporting y análisis temporal
- **Consistencia de Fechas:** Normalización automática a medianoche

Mapeo Integrado:

- **Una Sola Consulta:** Cliente y zona obtenidos simultáneamente
- **Zona por Defecto:** Manejo inteligente de clientes sin zona asignada
- **Validación Mejorada:** Verificación de existencia de columnas requeridas

Flexibilidad de Estados:

- **Detección Automática:** Maneja diferentes nombres de columna para estado
- **Robustez:** Adaptación a variaciones en el formato de entrada

Gestión de Recursos Mejorada:

- **Liberación Garantizada:** Bloque `finally` para cleanup de conexiones
- **Logging de Estado:** Confirmación de liberación de recursos

5. Mantenimiento y Solución de Problemas

Errores Comunes y Soluciones

Errores de Conexión:

```
Ocurrió un error inesperado en el script: [Error específico]
```

- **Causa:** Credenciales incorrectas, servidor inaccesible o base de datos no disponible
- **Solución:** Verificar archivo `.env`, conectividad y estado del servidor

Errores de Estructura de BD:

```
VERIFICA que la columna 'id_zone' exista en tu tabla 'Clientes'.
```

- **Causa:** Columna `id_zone` no existe en la tabla `Clientes`
- **Solución:** Verificar estructura de la tabla o modificar el script

Errores de Mapeo:

```
Advertencia: Los siguientes clientes no se encontraron y se omitirán...
```

- **Causa:** Nombres de clientes en CSV no coinciden con la base de datos
- **Solución:** Actualizar tabla `Clientes` o revisar función de limpieza

Errores de Inserción:

```
¡ERROR DURANTE LA INSERCIÓN!
Tipo de error: [Tipo específico]
```

- **Causa:** Violaciones de restricciones, tipos incompatibles o tabla bloqueada
- **Solución:** Verificar estructura de `Pending_Orders` y restricciones

Logs y Monitoreo

Información de Progreso:

- **Conexión:** Estado de conectividad con validación previa
- **Carga:** Confirmación de lectura exitosa del CSV
- **Procesamiento:** Confirmación de cada fase de transformación
- **Mapeo:** Resultados del proceso de enriquecimiento
- **Inserción:** Totales de registros procesados e insertados

Warnings y Alertas:

- **Clientes No Mapeados:** Lista específica con nombres exactos
- **Zona por Defecto:** Notificación de clientes con zona asignada automáticamente
- **Valores por Defecto:** Campos con valores de fallback aplicados

6. Consideraciones Técnicas

Optimizaciones de Rendimiento

Procesamiento Eficiente:

- **Pandas Vectorizado:** Operaciones masivas con `.fillna()`, `.astype()`
- **Regex Optimizado:** Función de limpieza con expresiones regulares eficientes
- **Single Pass Processing:** Minimización de iteraciones sobre los datos

Memory Management:

- **DataFrame Slicing:** Selección específica de columnas para reducir memoria
- **Copy Operations:** Uso controlado de `.copy()` solo cuando necesario
- **Resource Cleanup:** Liberación explícita de conexiones de BD

Robustez y Confiabilidad

Manejo de Datos Inconsistentes:

- **Fechas Inválidas:** Valor por defecto `1900-01-01` para fechas no convertibles
- **Valores Nulos:** Defaults específicos por tipo de campo
- **Tipos de Datos:** Conversiones seguras con manejo de errores

Validación de Integridad:

- **Foreign Keys:** Verificación de existencia de clientes antes de inserción
- **Restricciones de Longitud:** Limitación proactiva de campos de texto
- **Consistencia Temporal:** Normalización uniforme de fechas

Compatibilidad y Dependencias

Bibliotecas Principales:

- **pandas:** Manipulación y análisis de datos
- **numpy:** Operaciones numéricas eficientes
- **sqlalchemy:** ORM y manejo robusto de base de datos

- **pyodbc:** Driver complementario para SQL Server
- **datetime:** Manejo de fechas y timestamps
- **tkinter:** Interfaz de selección de archivos

7. Estrategia de Datos y Análisis

Concepto de Snapshot Diario para Órdenes

Filosofía de Diseño:

- **Estado Completo:** Cada ejecución captura todas las órdenes pendientes del momento
- **Evolución Temporal:** Permite análisis de cómo cambian las órdenes pendientes
- **Dimensiones Temporales:** Facilita análisis por mes, trimestre, año

Análisis Recomendado:

- **Órdenes Actuales:** Filtrar por `MAX(FechaCarga)` para estado más reciente
- **Tendencias de Pedidos:** Comparar volúmenes entre diferentes fechas de carga
- **Análisis Estacional:** Usar campos `mes` y `nombre_mes` para patrones estacionales

Casos de Uso de Análisis

Reportes de Estado Actual:

```sql
SELECT * FROM Pending_Orders
WHERE FechaCarga = (SELECT MAX(FechaCarga) FROM Pending_Orders)
```

Análisis de Tendencias por Mes:

```sql
SELECT nombre_mes, COUNT(*) as Total_Ordenes, SUM(amount_net) as Monto_Total
FROM Pending_Orders
WHERE FechaCarga = (SELECT MAX(FechaCarga) FROM Pending_Orders)
GROUP BY mes, nombre_mes
ORDER BY mes
```

Evolución de Órdenes Pendientes:

```sql
```

```sql
SELECT FechaCarga, COUNT(*) as Ordenes_Pendientes, SUM(amount_net) as Valor_Total
FROM Pending_Orders
GROUP BY FechaCarga
ORDER BY FechaCarga DESC
```

Análisis por Cliente y Zona:

```sql
SELECT c.nombre_cliente, p.id_zone, COUNT(*) as Ordenes, SUM(p.amount_net) as Total
FROM Pending_Orders p
JOIN Clientes c ON p.id_cliente = c.id_cliente
WHERE p.FechaCarga = (SELECT MAX(FechaCarga) FROM Pending_Orders)
GROUP BY c.nombre_cliente, p.id_zone
ORDER BY Total DESC
```

8. Roadmap y Mejoras Futuras

Versión Actual: 2.0

Características Principales:

- **Enriquecimiento temporal automático** con dimensiones completas
- **Mapeo integrado** de clientes y zonas en una operación
- **Flexibilidad de formatos** con detección automática de columnas
- **Gestión robusta de recursos** con cleanup garantizado

Mejoras Planificadas v2.1

Funcionalidades de Usuario:

- **Dashboard de Órdenes:** Visualización en tiempo real del estado de órdenes
- **Alertas de Volumen:** Notificaciones cuando el volumen de órdenes excede umbrales
- **Análisis Predictivo:** Estimaciones de fulfillment basadas en históricos

Mejoras Técnicas:

- **Validación de Calidad:** Reglas de negocio para detectar órdenes anómalas
- **Paralelización:** Procesamiento simultáneo de múltiples archivos
- **Cache Inteligente:** Reutilización de mapeos de clientes entre ejecuciones

Roadmap a Largo Plazo v3.0

Integración Avanzada:

- **API de Órdenes:** Interfaz REST para consultas en tiempo real
- **Integración ERP:** Conexión directa con sistemas de planificación
- **Workflows Automatizados:** Disparadores automáticos basados en eventos

Inteligencia de Negocio:

- **ML para Predicción:** Modelos para estimar tiempos de entrega
- **Optimización de Inventario:** Recomendaciones basadas en órdenes pendientes
- **Análisis de Rentabilidad:** Correlación entre órdenes y márgenes

9. Consideraciones de Seguridad y Cumplimiento

Protección de Datos Comerciales

- **Encriptación de Conexiones:** SQLAlchemy con conexiones seguras
- **Gestión de Credenciales:** Variables de entorno fuera del código
- **Auditoría de Accesos:** Logging de todas las operaciones de carga

Integridad de Datos Comerciales

- **Validación de Órdenes:** Verificación de consistencia de montos y cantidades
- **Trazabilidad Completa:** `FechaCarga` para auditoría total
- **Backup Automático:** Respaldo antes de cada carga masiva

Cumplimiento Operativo

- **Retención de Históricos:** Preservación de evolución de órdenes
- **Segregación de Responsabilidades:** Accesos diferenciados por rol
- **Continuidad de Negocio:** Estrategia de recuperación ante fallos

Este script representa una solución integral para el manejo de órdenes pendientes, proporcionando robustez, flexibilidad temporal y facilidad de análisis para la toma de decisiones operativas y estratégicas.