Documentación: Script de Carga de Cartera v2.0

Este documento detalla el funcionamiento, uso y mantenimiento del script ETL (Extracción, Transformación y Carga) diseñado para procesar archivos CSV con el estado de la cartera de clientes y cargarlos como una "foto diaria" completa en una base de datos SQL Server.

1. Funcionalidades Principales

El script automatiza la actualización de la tabla `Cartera` implementando una estrategia de snapshot diario con transformaciones específicas de datos financieros y lógica de negocio especializada.

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 Errores SQLAlchemy:** Captura específicamente errores de `SQLAlchemyError` para problemas de conectividad.

Selección y Procesamiento de Archivos Especializado

Carga Inteligente de CSV:

- **Filtros de Archivo:** Interfaz específica para archivos CSV de cartera
- **Omisión de Encabezados/Pie:** Utiliza `skiprows=6` y `skipfooter=1` para manejar formato específico del reporte de cartera
- **Motor Python:** Usa `engine='python'` para compatibilidad con `skipfooter`

Validación y Manejo de Errores:

- Verificación específica de `FileNotFoundError`
- Manejo robusto de errores con `sys.exit(1)` para códigos de error apropiados
- Logging detallado del progreso de carga

Transformación y Estandarización de Datos

Mapeo de Columnas Especializado:

Conversión de nombres de columnas del formato del sistema origen:

- `Zones for Financial Reporting` → `zona_csv_original`
- `Customer:Project` → `nombre_cliente`
- `Transaction Type` → `tipo_transaccion`
- `Date` → `fecha_facturacion`
- `Document Number` → `document_number`
- `Due Date` → `fecha_pago`
- `Open Balance` → `open_balance`

Limpieza de Datos:

- **Eliminación de Columnas:** Automáticamente elimina `P.O. No.` y `Age` por ser irrelevantes
- **Normalización de Nombres:** Función `clean_customer_name()` para estandarización robusta

Lógica de Negocio Específica para E-Commerce

Reglas de Transformación Automática:

El script implementa lógica especializada para clientes de comercio electrónico:

Walmart E-Commerce:

- **Condición:** `zona_csv_original == 'Walmart'` AND `nombre_cliente == 'Ecommerce'`
- **Transformación:**
  - `zona_csv_original` → `'E-Commerce'`
  - `nombre_cliente` → `'Walmart Ecommerce'`

Amazon:

- **Condición:** `zona_csv_original == 'Amazon'` AND `nombre_cliente == 'Ecommerce'`
- **Transformación:**
  - `zona_csv_original` → `'E-Commerce'`
  - `nombre_cliente` → `'Amazon'`

Manejo de Registros Sin Cliente:

- Conversión automática de `'- no customer/project -'` a `'Sin Nombre'`
- Preservación de integridad referencial

Procesamiento de Datos Financieros Avanzado

Transformación de `open_balance`:

Manejo específico de valores monetarios con formato contable:

1. **Conversión de Negativos:** `(valor)` → `-valor`
2. **Eliminación de Símbolos:** Remoción de `$`, `,` y espacios
3. **Conversión Numérica:** `pd.to_numeric()` con manejo de errores
4. **Valores por Defecto:** `fillna(0)` para valores no convertibles

Proceso Step-by-Step:

```python
'($1,234.56)' → '-$1,234.56' → '-1234.56' → -1234.56
'$5,678.90'   → '5678.90'    → 5678.90
```

Enriquecimiento de Datos con Base de Datos

Mapeo Dinámico Mejorado:

- **Consulta Completa:** Extrae `id_cliente`, `nombre_cliente` e `id_zone` de la tabla `Clientes`
- **Limpieza Bidireccional:** Aplica `clean_customer_name()` a ambos datasets
- **Merge Inteligente:** Utiliza `pd.merge()` con `how='left'` y sufijos específicos
- **Fallback de Zona:** Usa `zona_csv_original` cuando `id_zone` no está disponible

Validación de Integridad:

- **Conversión Segura:** `pd.to_numeric()` con `errors='coerce'` para `id_cliente`
- **Filtrado de No Mapeados:** Elimina registros sin `id_cliente` válido
- **Reporting Detallado:** Lista específica de clientes no encontrados

Estrategia de Snapshot Diario

Concepto de "Foto Diaria":

- **Carga Completa:** Todos los registros válidos del archivo se insertan
- **Marca Temporal:** Columna `FechaCarga` con `date.today()` para todos los registros
- **Sin Deduplicación:** Cada ejecución representa el estado completo del día

Ventajas del Snapshot:

- **Historia Completa:** Permite análisis de evolución temporal
- **Simplicidad:** No requiere lógica compleja de updates/inserts
- **Consistencia:** Estado completo y coherente por fecha

2. Arquitectura de Datos

Tabla de Destino: `Cartera`

Estructura esperada para recibir los datos procesados:

- `id_cliente` (INT, FK a tabla Clientes)
- `id_zone` (INT/VARCHAR, zona del cliente)
- `tipo_transaccion` (VARCHAR)
- `fecha_facturacion` (DATE, formato 'YYYY-MM-DD')
- `document_number` (VARCHAR)
- `fecha_pago` (DATE, formato 'YYYY-MM-DD')
- `open_balance` (DECIMAL/MONEY, valores negativos permitidos)
- `FechaCarga` (DATE, timestamp de la carga)

Tabla de Referencia: `Clientes`

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

Configuraciones de Base de Datos

```env
SERVER_NAME=nombre_servidor_sql
PORT=puerto_sql_server
DATABASE_NAME=nombre_base_datos
DB_USERNAME=usuario_base_datos
DB_PASSWORD=contraseña_usuario
```

3. Manejo y Uso del Script

Distribución

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

Requisitos Previos

1. Archivo de Configuración (.env)

Debe ubicarse en la misma carpeta que el ejecutable:

```
env

SERVER_NAME=servidor_cartera
PORT=1433
DATABASE_NAME=FinanzasDB
DB_USERNAME=usuario_cartera
DB_PASSWORD=password_seguro
```

2. Estructura de Base de Datos

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

3. Archivo CSV de Cartera

- **Formato:** Archivo CSV con estructura específica del sistema financiero
- **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 cartera desde el sistema financiero
- Guardar como archivo CSV con nombre descriptivo (ej: `cartera_2024-01-15.csv`)

2. Ejecución del Script

- Ejecutar el archivo `.exe`
- El sistema valida automáticamente la conexión a la base de datos:

```
Conexión a SQL Server 'FinanzasDB' en 'servidor_cartera' establecida.
```

3. Selección de Archivo

- Se presenta diálogo específico para archivos CSV:

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

4. Procesamiento Automático

La consola muestra el progreso detallado:

```
Archivo 'cartera_2024-01-15.csv' cargado exitosamente.
Columnas disponibles después de renombrar: [lista_columnas]
Limpiando y convirtiendo 'open_balance'...
'open_balance' procesado exitosamente.
id_cliente e id_zone mapeados exitosamente.
```

5. Carga por Lotes

```
Total de filas en el DataFrame de origen: 5000
Filas a insertar (snapshot diario completo): 4850
Iniciando inserción por lotes en la tabla 'Cartera'...
Lote insertado exitosamente: filas 0 a 1000
Lote insertado exitosamente: filas 1000 a 2000
...
Proceso de carga de 'Cartera' finalizado. Total de filas insertadas: 4850
```

4. Mejoras de la Versión 2.0

Diferencias vs Versión 1.0

Aspecto	Versión 1.0	Versión 2.0
Limpieza de Nombres	Básica	Función `clean_customer_name()` robusta
Mapeo de Clientes	Simple	Merge con limpieza bidireccional
Manejo de Zonas	Estático	Fallback dinámico con `zona_csv_original`

Aspecto	Versión 1.0	Versión 2.0
Validación de Errores	General	Específica por tipo (`SQLAlchemyError`)
Transformación Financiera	Básica	Procesamiento step-by-step robusto
Códigos de Salida	`exit()` genérico	`sys.exit(1)` con códigos específicos

Nuevas Funcionalidades v2.0

Limpieza de Datos Avanzada:

- **Regex Inteligente:** Eliminación de caracteres especiales preservando alfanuméricos
- **Normalización de Espacios:** Conversión de múltiples espacios a uno solo
- **Case Insensitive:** Conversión a minúsculas para comparaciones consistentes

Procesamiento Financiero Robusto:

- **Manejo de Paréntesis:** Conversión automática de formato contable a valores negativos
- **Eliminación de Símbolos:** Limpieza exhaustiva de símbolos monetarios
- **Validación Numérica:** Conversión segura con manejo de errores

Lógica de Negocio Especializada:

- **Reglas E-Commerce:** Transformaciones automáticas para Walmart y Amazon
- **Preservación de Contexto:** Mantenimiento de información de zona original
- **Flexibilidad:** Fácil extensión para nuevos casos de negocio

5. Mantenimiento y Solución de Problemas

Errores Comunes y Soluciones

Errores de Conexión:

```
Error de conexión a la base de datos: [SQLAlchemyError específico]
```

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

Errores de Estructura de Archivo:

```
Error: El archivo de entrada no se encontró...
```

- **Causa:** Archivo no seleccionado o ruta incorrecta
- **Solución:** Verificar ubicación del archivo y permisos de lectura

Errores de Mapeo de Clientes:

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

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

Errores de Inserción:

```
Error al insertar lote. Mensaje: [Error específico]
```

- **Causa:** Violación de restricciones, tipos de datos incompatibles o tabla bloqueada
- **Solución:** Verificar estructura de tabla y restricciones de integridad

Logs y Monitoreo

Información de Progreso:

- **Conexión:** Estado de conectividad a base de datos
- **Carga:** Confirmación de lectura exitosa del archivo
- **Transformación:** Resultados del procesamiento de `open_balance`
- **Mapeo:** Éxito en la asignación de IDs de cliente
- **Inserción:** Progreso detallado por lotes

Warnings y Alertas:

- **Clientes No Mapeados:** Lista específica con nombres exactos
- **Datos Filtrados:** Cantidad de registros omitidos por falta de `id_cliente`
- **Transformaciones Aplicadas:** Confirmación de reglas de negocio ejecutadas

6. Consideraciones Técnicas

Optimizaciones de Rendimiento

Procesamiento Vectorizado:

- **NumPy Operations:** Uso de `np.where()` para transformaciones condicionales masivas

- **Pandas Efficiency:** Operaciones vectorizadas para limpieza de datos
- **Batch Processing:** Inserción en lotes de 1000 registros

Memory Management:

- **DataFrame Copying:** Uso juicioso de `.copy()` para evitar SettingWithCopyWarning
- **Column Dropping:** Eliminación temprana de columnas innecesarias
- **Data Type Conversion:** Conversiones eficientes con manejo de errores

Robustez y Confiabilidad

Manejo de Valores Nulos:

- **Open Balance:** Conversión de valores no numéricos a 0
- **Fechas:** Manejo de errores con `errors='coerce'` en `pd.to_datetime()`
- **IDs de Cliente:** Filtrado de registros sin mapeo válido

Transacciones:

- **Begin Block:** Todas las inserciones dentro de una transacción
- **Rollback Automático:** Reversión ante cualquier error
- **Consistencia:** Estado de base de datos garantizado

Compatibilidad y Dependencias

Bibliotecas Principales:

- **pandas:** Manipulación y análisis de datos
- **numpy:** Operaciones numéricas eficientes
- **sqlalchemy:** ORM y manejo de base de datos
- **pyodbc:** Driver complementario para SQL Server
- **python-dotenv:** Gestión de variables de entorno
- **tkinter:** Interfaz de selección de archivos

Compatibilidad de Sistemas:

- **Windows:** Ejecutable PyInstaller nativo
- **SQL Server:** Compatibilidad completa con versions modernas
- **Python:** Requiere Python 3.6+ para desarrollo

7. Estrategia de Datos y Análisis

Concepto de Snapshot Diario

Filosofía de Diseño:

- **Estado Completo:** Cada ejecución captura el estado total de la cartera
- **Histórico Preservado:** Mantiene evolución temporal de la cartera
- **Simplicidad Operativa:** Sin lógica compleja de sincronización

Análisis Recomendado:

- **Última Fecha:** Filtrar por `MAX(FechaCarga)` para estado actual
- **Tendencias:** Comparar snapshots entre diferentes fechas
- **Evolución:** Análizar cambios en `open_balance` por cliente/zona

Casos de Uso de Análisis

Reportes Diarios:

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

Análisis de Tendencias:

```sql
SELECT FechaCarga, SUM(open_balance) as Total_Cartera
FROM Cartera
GROUP BY FechaCarga
ORDER BY FechaCarga DESC
```

Comparación Mensual:

```sql
```

```
SELECT
    id_cliente,
    open_balance as Saldo_Actual,
    LAG(open_balance) OVER (PARTITION BY id_cliente ORDER BY FechaCarga) as Saldo_Anterior
FROM Cartera
WHERE FechaCarga IN (SELECT DISTINCT TOP 2 FechaCarga FROM Cartera ORDER BY FechaCarga DESC)
```

8. Roadmap y Mejoras Futuras

Versión Actual: 2.0

Características Principales:

- **Procesamiento robusto** de datos financieros
- **Lógica de negocio especializada** para E-Commerce
- **Mapeo dinámico mejorado** con fallbacks
- **Validación exhaustiva** de datos y conexiones

Mejoras Planificadas v2.1

Funcionalidades de Usuario:

- **Preview de Datos:** Vista previa del snapshot antes de carga
- **Validación de Calidad:** Reportes de calidad de datos automáticos
- **Interfaz de Configuración:** Editor para reglas de negocio

Mejoras Técnicas:

- **Logging a Archivo:** Sistema de logs persistente con rotación
- **Métricas de Performance:** Tiempo de procesamiento por fase
- **Backup Automático:** Respaldo antes de cada carga

Roadmap a Largo Plazo v3.0

Análisis Integrado:

- **Dashboard Web:** Portal de análisis de cartera en tiempo real
- **Alertas Inteligentes:** Notificaciones por cambios significativos
- **API REST:** Interfaz programática para integración

Inteligencia de Negocio:

- **Predicción de Cobros:** Modelos ML para estimación de recuperación
- **Detección de Anomalías:** Identificación automática de irregularidades
- **Optimización de Cobranza:** Recomendaciones basadas en históricos

9. Consideraciones de Seguridad y Cumplimiento

Protección de Datos Financieros

- **Encriptación en Tránsito:** Conexiones SQLAlchemy seguras
- **Variables de Entorno:** Credenciales fuera del código fuente
- **Logs Sanitizados:** Sin exposición de información sensible

Auditoría y Trazabilidad

- **Timestamp Completo:** `FechaCarga` para trazabilidad total
- **Integridad Referencial:** Validación de foreign keys
- **Consistencia Transaccional:** Operaciones ACID completas

Cumplimiento Normativo

- **Retención de Datos:** Historial completo para auditorías
- **Segregación de Accesos:** Permisos específicos por usuario
- **Backup y Recuperación:** Estrategia de continuidad de negocio

Este script representa una solución robusta y especializada para el manejo de datos de cartera, proporcionando confiabilidad, trazabilidad y flexibilidad para análisis financiero avanzado.