### **Descripción del código**

Este código tiene como objetivo monitorear una carpeta específica para detectar la creación de nuevos archivos CSV. Cuando un archivo CSV es detectado, el código lee su contenido y, dependiendo del nombre del archivo, inserta los datos en una tabla correspondiente de una base de datos SQL Server. A continuación, se proporciona un análisis detallado del código.

### **Descripción general**

1. **Dependencias**: El código utiliza varias bibliotecas de Python:
   * time: Para realizar pausas o dormir durante la ejecución del código.
   * pyodbc: Para realizar conexiones a bases de datos SQL Server utilizando ODBC.
   * watchdog: Para monitorear cambios en un sistema de archivos (carpeta específica).
   * pandas: Para leer y manipular los archivos CSV.
2. **Estructura del código**: El código está organizado en varias partes:
   * **Clase FileHandler**: Maneja los eventos relacionados con el sistema de archivos (cuando se crea un nuevo archivo en la carpeta monitoreada).
   * **Función process\_file**: Procesa los archivos CSV y decide en qué tabla de la base de datos insertar los datos.
   * **Métodos de inserción**: Métodos para insertar los datos de cada archivo CSV en la base de datos, dependiendo de la tabla a la que se refiera el archivo.
   * **Función start\_watching**: Inicializa la observación de la carpeta y gestiona la conexión a la base de datos.

### **Análisis detallado**

1. **Monitoreo de archivos con watchdog**:
   * Se utiliza la librería watchdog para observar la carpeta que se pasa como parámetro (folder\_to\_watch).
   * Se crea un **Observer**, que continuamente verifica si hay cambios en la carpeta, y un **FileSystemEventHandler** que maneja los eventos cuando se detecta un nuevo archivo.
   * El evento que se maneja es **on\_created**, que se activa cuando se crea un nuevo archivo. El código verifica si el archivo es un archivo CSV, y en caso afirmativo, lo procesa.
2. **Procesamiento de archivos CSV**:
   * Cuando se detecta un archivo CSV, el código lee el archivo utilizando pandas.read\_csv().
   * Después de leer el archivo, el nombre del archivo se utiliza para determinar a qué tabla de la base de datos se deben insertar los datos. El nombre del archivo se usa sin la extensión .csv y se convierte a minúsculas para hacer la comparación (por ejemplo, aircrafts.csv -> tabla aircrafts).
   * Si el nombre del archivo coincide con una de las tablas definidas (por ejemplo, aircrafts, airports, flights, etc.), el código llama a un método específico para insertar los datos en esa tabla.
3. **Métodos de inserción en SQL Server**: El código tiene métodos específicos para insertar los datos en diferentes tablas de la base de datos. Cada uno de estos métodos hace lo siguiente:
   * Establece una conexión con la base de datos.
   * Itera sobre cada fila del archivo CSV utilizando df.iterrows() (proporcionado por pandas).
   * Utiliza una sentencia INSERT INTO para insertar los datos en las tablas correspondientes.
   * Después de insertar los datos, realiza un commit para asegurar que los cambios se guarden en la base de datos.
4. Los métodos de inserción son los siguientes:
   * insert\_into\_aircrafts: Inserta datos en la tabla aircrafts\_data.
   * insert\_into\_airports: Inserta datos en la tabla airports\_data.
   * insert\_into\_flights: Inserta datos en la tabla flights.
   * insert\_into\_boarding\_passes: Inserta datos en la tabla boarding\_passes.
   * insert\_into\_booking: Inserta datos en la tabla bookings.
   * insert\_into\_seats: Inserta datos en la tabla seats.
   * insert\_into\_tickets: Inserta datos en la tabla tickets.
   * insert\_into\_ticket\_flights: Inserta datos en la tabla ticket\_flights.
5. **Eliminación de archivos procesados**: Después de procesar e insertar los datos en la base de datos, el archivo CSV es eliminado con os.remove(file\_path). Esto asegura que el archivo no se procese nuevamente.
6. **Conexión a SQL Server**: La conexión a la base de datos SQL Server se realiza mediante pyodbc.connect(). La cadena de conexión utilizada en el código está configurada con parámetros básicos de SQL Server, como el nombre del servidor y la base de datos. El uso de Trusted\_Connection=yes permite la autenticación de Windows para la conexión.
7. **Manejo de excepciones**: El código maneja de manera simple la detención del observador si el proceso es interrumpido manualmente con KeyboardInterrupt. Esto se hace mediante el bloque try...except.

### **Función principal**

* **start\_watching(directory\_to\_watch, db\_connection\_string)**: Esta función se encarga de inicializar la conexión a la base de datos, crear el manejador de eventos, y comenzar a observar la carpeta para detectar los archivos CSV.
* El ciclo while True mantiene la observación activa hasta que se interrumpe manualmente (por ejemplo, mediante Ctrl+C).

### **Recomendaciones y mejoras**

1. **Manejo de errores**:
   * Actualmente, el código no tiene manejo de errores robusto. Sería beneficioso agregar manejo de excepciones, especialmente en las operaciones de base de datos y al procesar los archivos. Por ejemplo, si el formato de un CSV es incorrecto o si hay problemas con la conexión a la base de datos, el código debería manejar estos errores y registrar los detalles.
2. **Eficiencia en la inserción de datos**:
   * En lugar de insertar fila por fila, se podría mejorar la eficiencia utilizando métodos como executemany de pyodbc, que permite insertar varias filas de manera más eficiente en una sola ejecución.
3. **Validación de datos**:
   * Sería útil agregar una validación de los datos antes de insertarlos en la base de datos para asegurarse de que los datos en los archivos CSV sean válidos y completos. Esto puede evitar errores o datos incorrectos en la base de datos.
4. **Registro de eventos**:
   * El código podría beneficiarse de una mejor implementación de registros (logging) para poder rastrear eventos, como archivos procesados, errores, y cualquier problema que ocurra durante la ejecución.
5. **Configuración dinámica de las tablas**:
   * Actualmente, el código asume que los archivos CSV coinciden con las tablas de base de datos predeterminadas. Sería útil hacer que la configuración de las tablas sea más flexible o configurable, de modo que pueda adaptarse a diferentes nombres de archivos o bases de datos sin necesidad de modificar el código fuente.

### **Conclusión**

El código es una solución efectiva para automatizar el proceso de carga de datos desde archivos CSV hacia una base de datos SQL Server. Utiliza bibliotecas como pandas para leer los archivos CSV, pyodbc para la conexión a la base de datos, y watchdog para monitorear los cambios en una carpeta. Sin embargo, se pueden realizar mejoras en cuanto a manejo de errores, eficiencia y validación de datos para hacerlo más robusto y eficiente en un entorno de producción.