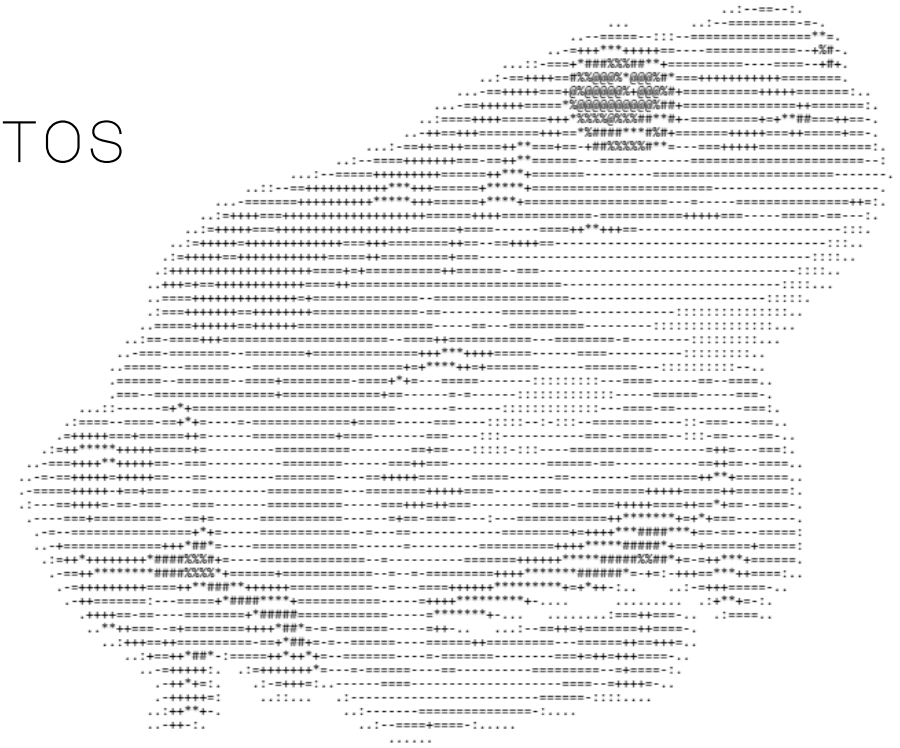
PROYECTOS
DE
SQL



Objetivo del Documento

El objetivo principal de este documento es introducir y profundizar en el manejo, almacenamiento y estructuración de datos dentro del entorno de desarrollo en Python, con especial énfasis en la utilización de la biblioteca SQLite (a través del módulo sqlite3) y otras herramientas complementarias que facilitan la gestión eficiente de la información.

Se busca brindar una comprensión integral del proceso de conexión, consulta, inserción y manipulación de bases de datos relacionales, al mismo tiempo que se incorporan librerías auxiliares —como pandas, SQLAlchemy u otras pertinentes— para optimizar el flujo de trabajo y garantizar la interoperabilidad entre estructuras tabulares y motores de base de datos.

En este marco, el documento pretende sentar las bases para el desarrollo de aplicaciones orientadas al análisis financiero o la automatización de procesos, donde la persistencia y accesibilidad de los datos resulta crítica para la trazabilidad y escalabilidad de los sistemas.

Proyecto I: Creación y manejo de bases de datos

Introducción

El objetivo de este primer proyecto es explorar y aplicar las funcionalidades principales que ofrece la biblioteca sqlite3 de Python —comúnmente referida como SQL en el entorno del lenguaje— para la gestión de bases de datos relacionales. A través de su implementación, se busca establecer una base sólida en la creación, administración, carga, consulta y eliminación de registros, elementos fundamentales para garantizar un almacenamiento de datos eficiente y estructurado.

La comprensión adecuada de estos procesos resulta esencial en el desarrollo de sistemas orientados al análisis financiero y, especialmente, en la construcción de bots de trading. En ambos casos, el acceso a datos históricos y actualizados es un componente crítico tanto para el funcionamiento operativo como para la optimización de modelos y estrategias. Al contar con una estructura de almacenamiento robusta y flexible, se facilita la trazabilidad de resultados, la reutilización de información y la automatización de procesos analíticos.

Este proyecto se presenta, por lo tanto, como una instancia introductoria pero fundamental dentro del enfoque general del desarrollo de herramientas cuantitativas en Python, sirviendo como soporte estructural para los módulos que se abordarán posteriormente.

A lo largo del desarrollo de este proyecto, se buscará implementar una base de datos que represente, de forma preliminar, las características estructurales que podrán adoptar los sistemas de almacenamiento utilizados por los bots de trading que se desarrollarán posteriormente. En esta etapa inicial, el contenido específico de la base de datos no constituye un aspecto central, ya que el enfoque estará orientado al reconocimiento y aplicación práctica de las funcionalidades que ofrece SQL en el entorno de Python.

El objetivo es familiarizarse con las operaciones fundamentales tales como la creación de tablas, inserción de datos, consultas, actualizaciones y eliminaciones, independientemente del dominio temático de la información. Esta aproximación abstracta permitirá construir una comprensión operativa de la gestión de datos relacionales, que luego podrá adaptarse con facilidad a los distintos requerimientos de los sistemas financieros automatizados en etapas futuras del proyecto.

Parte I: Creación de la base.

1.1 Los datos.

La creación de una base de datos en el contexto del desarrollo de herramientas de análisis financiero y bots de trading puede realizarse a partir de múltiples fuentes de información. Entre los métodos más utilizados se encuentran la descarga de datos mediante APIs especializadas como yfinance (para obtener datos bursátiles históricos y en tiempo real), el uso de técnicas de web scraping para extraer información de portales gubernamentales o sitios web financieros, y la carga de archivos en formato .csv previamente estructurados.

No obstante, en esta instancia introductoria, se optará por la generación manual de una base de datos simulada con el fin de comprender de manera clara y didáctica la estructura interna de las tablas, los tipos de datos utilizados y las relaciones que podrían establecerse entre las distintas entidades. Este enfoque permitirá centrarse en el funcionamiento de las operaciones básicas de SQL sin depender de fuentes externas o estructuras de datos complejas.

A continuación, se presenta una estructura de datos simulada en formato de lista de diccionarios en Python, la cual servirá como base para la posterior creación de una tabla en una base de datos relacional. Esta tabla representa una colección de Agentes de Liquidación y Compensación (ALyC), incluyendo información clave sobre su conexión al mercado, horarios operativos y rutas a los datos que podrían ser utilizados por un bot de trading:

```
data = [
   {
      "id": 1,
      "alyc": "InvertirOnline",
      "market": "MERV",
      "coneccion": "activa",
      "hora_inicio": "11",
      "hora_finalizacion": "17",
      "carpeta_de_datos": "xxxx/xxxx/xxxx",
      "id_datos_mercado": "/informacion_de_mercado",
      "id_datos_ordenes": "/informacion_de_ordenes"
   }
]
```

Este conjunto de datos será utilizado para poblar manualmente una tabla dentro de una base SQLite, permitiendo evaluar el uso de sentencias CREATE, INSERT INTO y posteriores operaciones de consulta y manipulación que serán exploradas en las próximas secciones.

1.2 Estructura y almacenamiento de datos.

Una vez definida la estructura de datos simulada, el siguiente paso consiste en transformarla en un DataFrame utilizando la biblioteca pandas, para posteriormente almacenarla como una tabla relacional en una base de datos SQLite. Este procedimiento permite trabajar con datos estructurados de forma persistente, facilitando su manipulación, consulta y explotación futura por parte de herramientas analíticas o algoritmos automatizados de trading.

El flujo de trabajo implementado puede describirse en los siguientes pasos:

- Conversión de la estructura de datos a un DataFrame: Se utiliza pandas.DataFrame para convertir la lista de diccionarios previamente definida en una estructura tabular.
- Conexión a una base de datos SQLite: A través de sqlite3.connect, se establece una conexión con una base de datos denominada config.db. En caso de que esta no exista, será creada automáticamente en el directorio de trabajo.
- Exportación del DataFrame a una tabla SQL: Se utiliza el método to_sql de pandas para almacenar el contenido del DataFrame en una tabla denominada config. El parámetro if_exists="replace" garantiza que, si la tabla ya existe, será reemplazada por la nueva información, y index=False evita que se inserte una columna adicional con los índices del DataFrame.
- Cierre de la conexión: Finalmente, se cierra la conexión a la base de datos con conn.close() para liberar recursos.

```
df_config = pd.DataFrame(data)

conn = sqlite3.connect("config.db")

df_config.to_sql("config", conn, if_exists="replace", index=False)

conn.close()
```

Con esta implementación, se concluye exitosamente la creación de la base de datos inicial, la cual servirá como punto de partida para las operaciones y experimentaciones posteriores. A partir de esta estructura, se podrán realizar consultas, modificaciones y análisis que permitirán evaluar las funcionalidades principales de la biblioteca SQL en un entorno controlado y representativo del contexto real de uso.

Bibliografía

Beaulieu, A. (2020). *Learning SQL*. O'Reilly Media.

Python Software Foundation. (n.d.). *sqlite3 — DB-API 2.0 interface for SQLite databases*. Python 3.12. https://docs.python.org/3/library/sqlite3.html