

Datos con Pandas II

(Teórico)



Introducción

Bienvenidos al módulo 20 - Datos con Pandas II. En el mismo, los objetivos de aprendizaje estarán orientados a incorporar conceptos sobre conexión a gestores de bases de datos relacionales (RDBMS) utilizando Python y también diferentes conceptos sobre el tratamiento de datasets para obtener subconjuntos de datos

Una vez finalizado este módulo estarás capacitado para las siguientes acciones:

- Conectarte a los RDBMS: SQLite, MySQL, PostgreSQL y SQL Server, utilizando Python.
- Interactuar con los RDBMS y ejecutar consultas SQL básicas.
- Realizar filtrados en dataframes incorporando diferentes técnicas y conceptos para realizar esa tarea.
- Interactuar con diferentes notebooks, que te permitirán reproducir todos los casos que cubre este módulo



Tema 1. Conexión a bases de datos con Python

Objetivo

Una vez finalizado este tema estarás capacitado para las siguientes acciones:

- Conectarte a los RDBMS: SQLite, MySQL, PostgreSQL y SQL Server, utilizando Python.
- Interactuar con los RDBMS y ejecutar consultas SQL básicas.
- Utilizar el ORM SQLAlchemy y así poder utilizar Pandas para interactuar con los RDBMS.

Introducción

Los sistemas de administración de bases de datos relacionales (RDBMS - Relational Database Management Systems) proporcionan una manera fácil de almacenar grandes cantidades de información. Además, con la ayuda de SQL podemos acceder, mantener y modificar los datos almacenados en ellos.

Si bien SQL nos permite realizar uniones o utilizar funciones de agregación, no proporciona los medios para realizar técnicas más avanzadas, como realizar pruebas estadísticas o crear modelos predictivos. Si deseamos realizar este tipo de operaciones, necesitaremos la ayuda de Python.

Esto plantea la pregunta: ¿cómo implementamos el código Python en bases de datos que responden a consultas SQL?.

Adaptador de base de datos

Para acceder a las bases de datos con Python, deberemos utilizar un adaptador de base de datos .

Python ofrece adaptadores de base de datos a través de sus módulos que permiten el acceso a las principales bases de datos como MySQL, PostgreSQL, SQL Server y SQLite.



Todos estos módulos se basan en la [PEP 249](#), especificación de API de base de datos de Python (DB-API) para administrar bases de datos. Como resultado, el código utilizado para administrar las bases de datos es coherente en todos los adaptadores de bases de datos.

Aprovechar los adaptadores de bases de datos requiere una comprensión de los objetos y métodos clave que se utilizarán para facilitar la interacción con la base de datos en cuestión.

Mapecto relacional de objetos

El mapeo objeto relacional de objetos (más conocido por su nombre en inglés, Object-Relational mapping, o sus siglas ORM) es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional como motor de persistencia.

En la práctica esto crea una base de datos orientada a objetos virtual, sobre la base de datos relacional que posibilita el uso de las características propias de la orientación a objetos (básicamente herencia y polimorfismo). Hay paquetes comerciales y de uso libre

disponibles que desarrollan el mapeo relacional de objetos, aunque algunos programadores prefieren crear sus propias herramientas ORM.

SQLAlchemy

SQLAlchemy es un ORM (Mapeador de relaciones de objetos), que está escrito en Python para trabajar con bases de datos. Ayuda a los programadores y desarrolladores de aplicaciones a tener una flexibilidad de control total sobre las herramientas SQL.

A menudo, al desarrollar aplicaciones en cualquier lenguaje de programación, nos encontramos con la necesidad de almacenar y leer datos de las bases de datos. Este módulo proporciona una forma Pythonic de crear y representar bases de datos relacionales desde dentro de los proyectos de Python. Una ventaja de trabajar con un módulo de este tipo es que no necesitamos recordar las diferencias sintácticas de las distintas bases de datos. El módulo hace todo el trabajo pesado por nosotros, mientras interactúa con todas las bases de datos de la misma manera.

Conexión a sistemas de gestión de bases de datos relacionales

A continuación realizaremos una breve descripción de los sistemas de gestión de bases de datos relaciones que luego vamos a conectar utilizando Python.

SQLite3

SQLite es una biblioteca C que proporciona una base de datos liviana basada en disco que no requiere un proceso de servidor separado y permite acceder a la base de datos utilizando una variante no estándar del lenguaje de consulta SQL.

Algunas aplicaciones pueden usar SQLite para el almacenamiento interno de datos. También es posible crear un prototipo de una aplicación usando SQLite y luego transferir el código a una base de datos más grande como PostgreSQL u Oracle.

SQLite viene integrado en la instalación de Python, por lo que no es necesario instalar ningún software adicional.

En el siguiente [link](#) vamos a encontrar un notebook con los pasos que debemos seguir para conectarnos a SQLite utilizando Python. Asimismo, utilizando esa conexión ejecutaremos algunas sentencias DDL y DML de SQL. Finalmente utilizaremos SQLAlchemy para crear un engine y trabajar con la base de datos utilizando Pandas

SQLite3 + SQL Alchemy + Pandas

```
In [1]: # Import main libraries
import sqlite3
import pandas as pd
import sqlalchemy as db
```

Pasos generales para trabajar con SQLite 3

1. **Crear una conexión a la base de datos:** utilizando el método `connect()` generamos un objeto de conexión a base de datos SQLite. Si es la primera vez que nos conectamos, se nos creará la base de datos en el mismo directorio donde se este ejecutando el notebook o archivo .py.
2. **Crear un cursor:** El método `cursor()` se utiliza para realizar la conexión y ejecutar consultas SQL que nos permiten crear tablas, insertar datos, etc. Para crear un cursor solo necesitamos usar la conexión que ya hemos creado.
3. **Ejecutar una sentencia SQL:** Una vez creado el cursor, podremos ejecutar las sentencias SQL utilizando el método `execute()`.
4. **Realizar un commit:** El método `commit()` de SQLite se utiliza para guardar cualquier transacción de forma permanente en el sistema de base de datos. Todas las modificaciones de datos o del sistema realizadas por el comando COMMIT desde el comienzo de las transacciones son de naturaleza permanente y no se pueden deshacer ni revertir, ya que una operación COMMIT exitosa libera todos los recursos de transacción involucrados.
5. **Desconectarnos de la base de datos:** El método `close()` de SQLite cierra la conexión a la base de datos.

Configuraciones principales

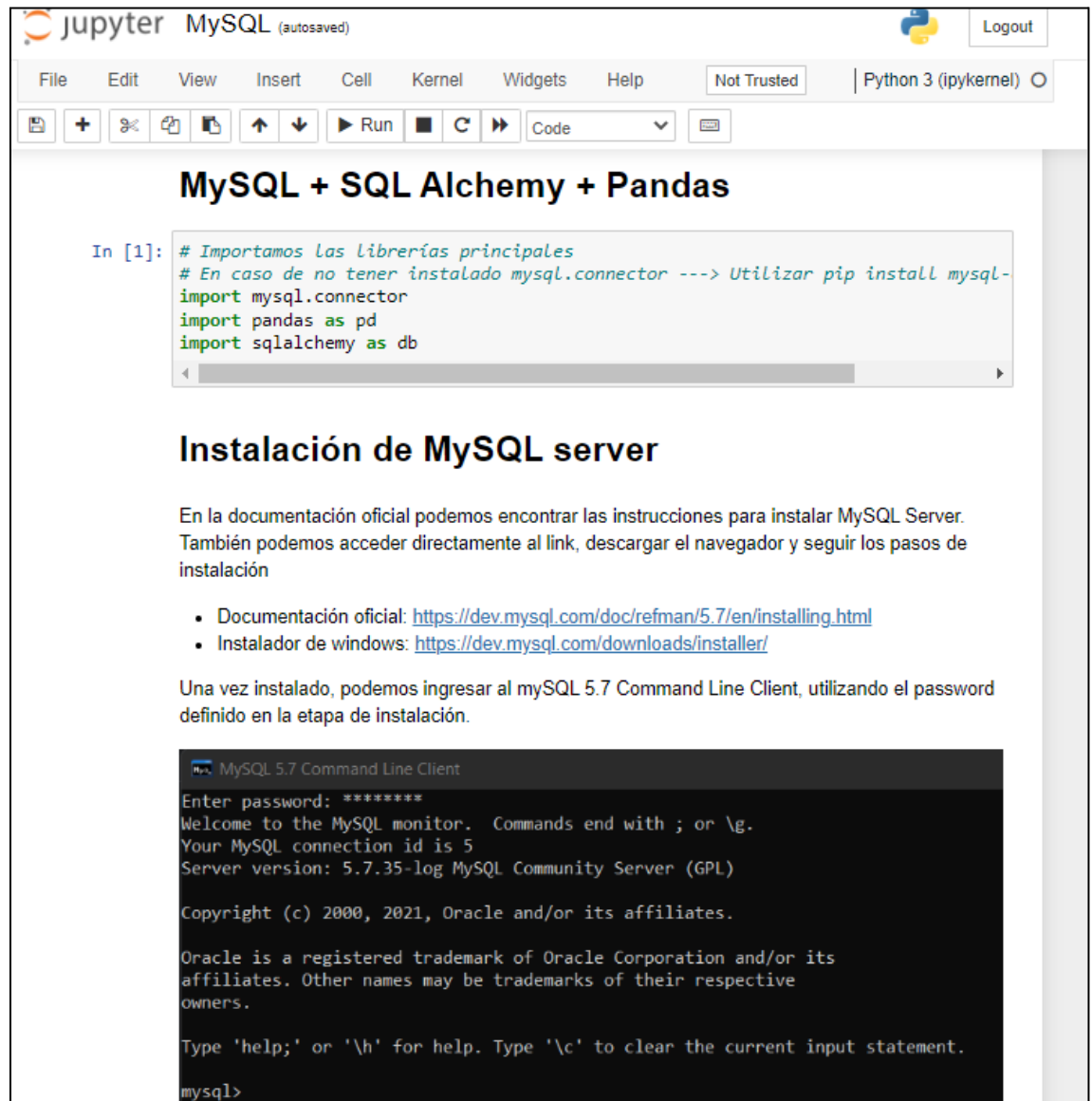
```
In [2]: # Crear una conexión a La base de datos
conn = sqlite3.connect('commerce.db')

# Crear un cursor
cursor = conn.cursor()
```

MySQL

MySQL es uno de los sistemas de gestión de bases de datos relacionales (RDBMS) más populares del mercado actual. Ocupó el segundo lugar después de Oracle DBMS en el [DB-Engines Ranking](#) del año 2022 . Como la mayoría de las aplicaciones de software necesitan interactuar con los datos de alguna forma, los lenguajes de programación como Python proporcionan herramientas para almacenar y acceder a estas fuentes de datos.

En el siguiente [link](#) vamos a encontrar un notebook con los pasos que debemos seguir para conectarnos a MySQL utilizando Python. Asimismo, utilizando esa conexión ejecutaremos algunas sentencias DDL y DML de SQL. Finalmente utilizaremos SQLAlchemy para crear un engine y trabajar con la base de datos utilizando Pandas.



MySQL + SQL Alchemy + Pandas

```
In [1]: # Importamos las librerías principales
# En caso de no tener instalado mysql.connector ---> Utilizar pip install mysql-
import mysql.connector
import pandas as pd
import sqlalchemy as db
```

Instalación de MySQL server

En la documentación oficial podemos encontrar las instrucciones para instalar MySQL Server. También podemos acceder directamente al link, descargar el navegador y seguir los pasos de instalación

- Documentación oficial: <https://dev.mysql.com/doc/refman/5.7/en/installing.html>
- Instalador de windows: <https://dev.mysql.com/downloads/installer/>

Una vez instalado, podemos ingresar al mySQL 5.7 Command Line Client, utilizando el password definido en la etapa de instalación.

```
MySQL 5.7 Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.7.35-log MySQL Community Server (GPL)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

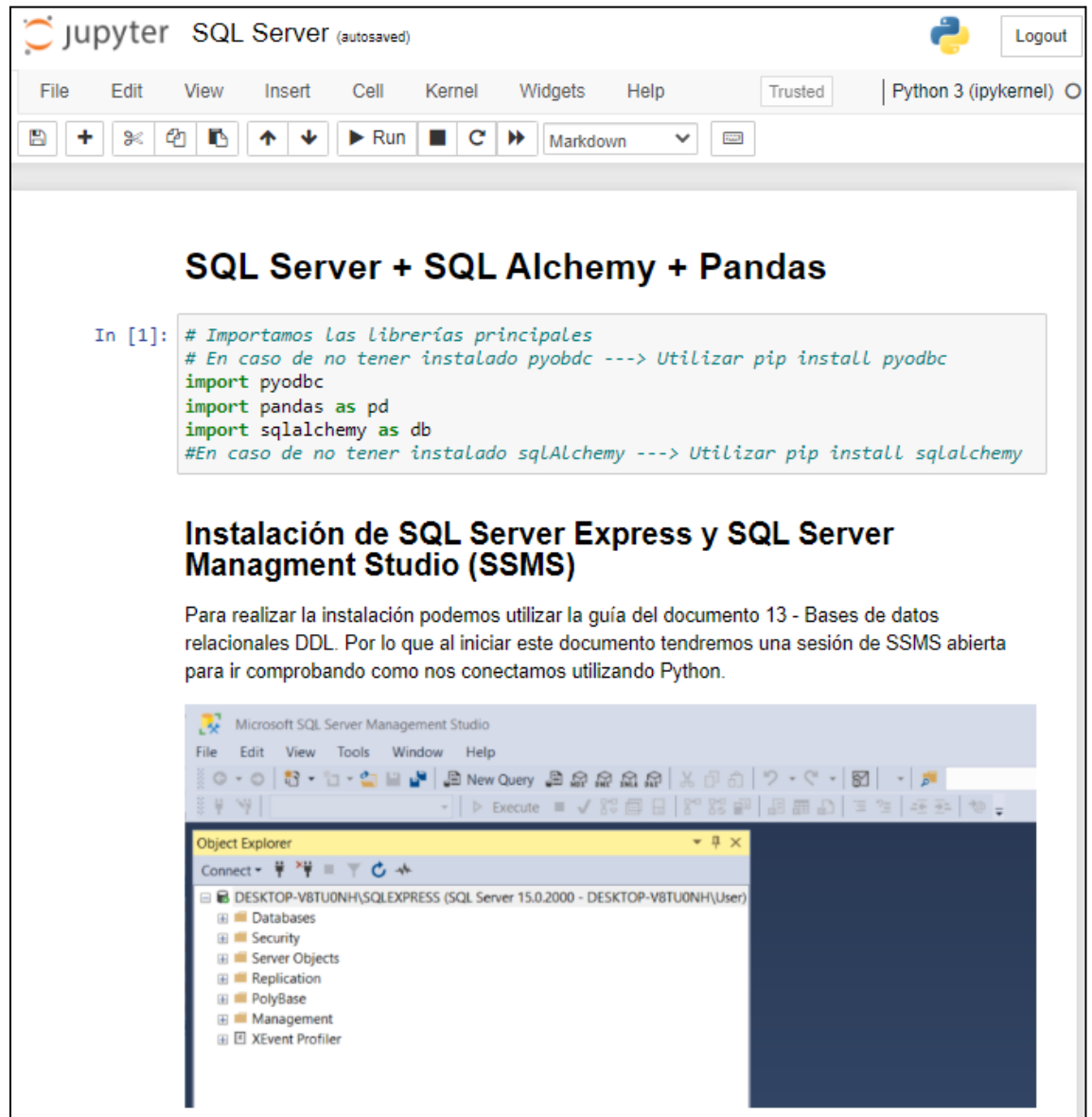
SQL Server

Microsoft SQL Server es un sistema de gestión de base de datos relacionales (RDBMS) desarrollado por la empresa Microsoft.

El lenguaje de desarrollo utilizado (por línea de comandos o mediante la interfaz gráfica de Management Studio) es Transact-SQL (TSQL), una implementación del estándar ANSI del lenguaje SQL, utilizado para manipular y recuperar datos (DML), crear tablas y definir relaciones entre ellas (DDL).

En el siguiente [link](#) vamos a encontrar un notebook con los pasos que debemos seguir para conectarnos a SQL Server utilizando Python.

Asimismo, utilizando esa conexión ejecutaremos algunas sentencias DDL y DML de SQL. Finalmente utilizaremos SQLAlchemy para crear un engine y trabajar con la base de datos utilizando Pandas.



SQL Server + SQL Alchemy + Pandas

```
In [1]: # Importamos las Librerías principales
# En caso de no tener instalado pyodbc ---> Utilizar pip install pyodbc
import pyodbc
import pandas as pd
import sqlalchemy as db
#En caso de no tener instalado sqlalchemy ---> Utilizar pip install sqlalchemy
```

Instalación de SQL Server Express y SQL Server Managment Studio (SSMS)

Para realizar la instalación podemos utilizar la guía del documento 13 - Bases de datos relacionales DDL. Por lo que al iniciar este documento tendremos una sesión de SSMS abierta para ir comprobando como nos conectamos utilizando Python.

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The 'Object Explorer' pane on the left displays the server structure for 'DESKTOP-V8TU0NH\SQLEXPRESS (SQL Server 15.0.2000 - DESKTOP-V8TU0NH\User)'. The tree view includes folders for Databases, Security, Server Objects, Replication, PolyBase, Management, and XEvent Profiler.

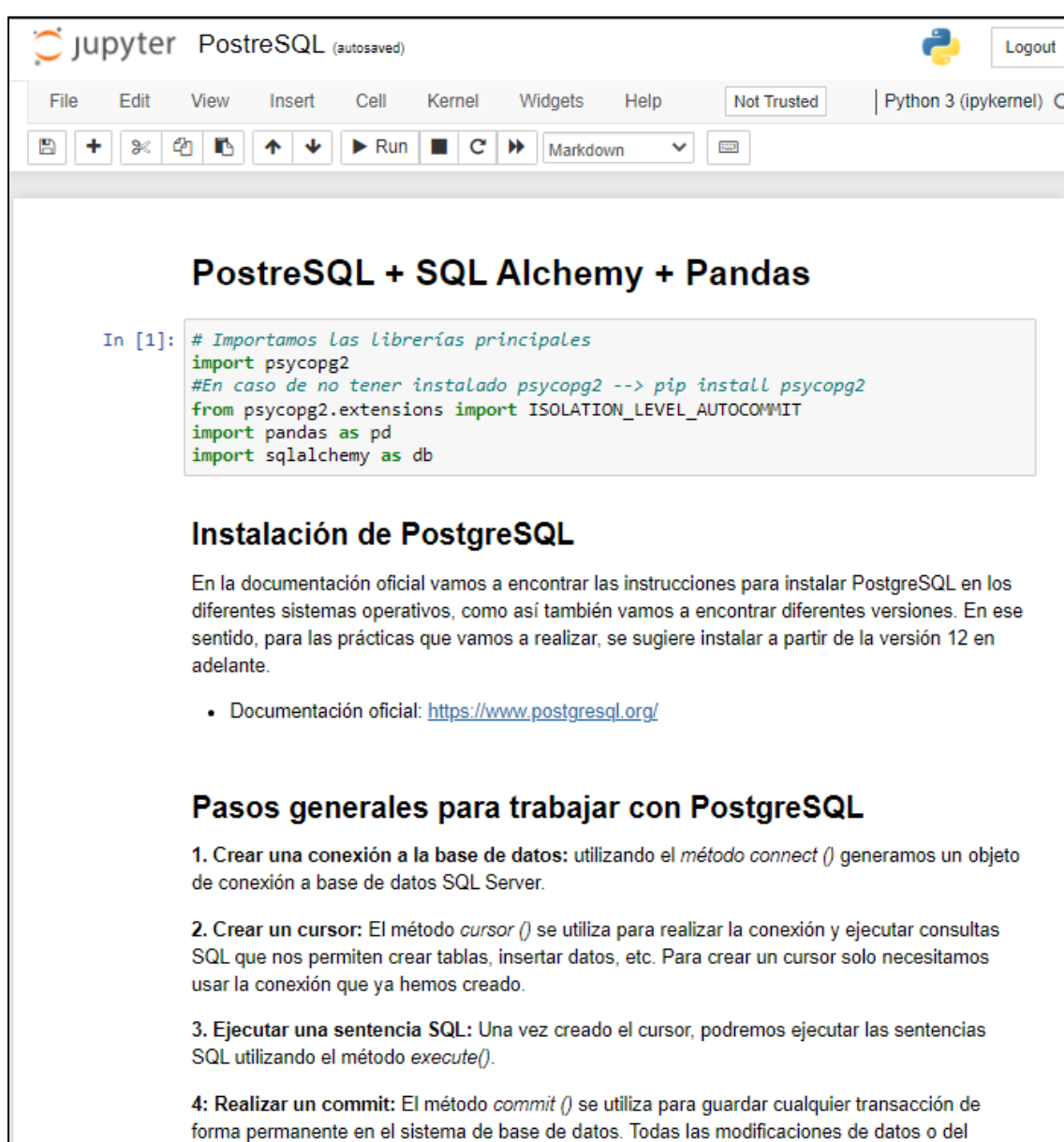
PostgreSQL

PostgreSQL es un sistema de gestión de base de datos relacionales (RDBMS) avanzado, de clase empresarial y de código abierto. Admite consultas SQL (relacionales) y JSON (no relacionales).

PostgreSQL es una base de datos altamente estable respaldada por más de 20 años de desarrollo por parte de la comunidad de código

abierto. Utiliza como base de datos principal para muchas aplicaciones web, así como para aplicaciones móviles y de análisis.

En el siguiente [link](#) vamos a encontrar un notebook con los pasos que debemos seguir para conectarnos a MySQL utilizando Python. Asimismo, utilizando esa conexión ejecutaremos algunas sentencias DDL y DML de SQL. Finalmente utilizaremos SQLAlchemy para crear un engine y trabajar con la base de datos utilizando Pandas.



PostgreSQL + SQL Alchemy + Pandas

```
In [1]: # Importamos las librerías principales
import psycopg2
#En caso de no tener instalado psycopg2 --> pip install psycopg2
from psycopg2.extensions import ISOLATION_LEVEL_AUTOCOMMIT
import pandas as pd
import sqlalchemy as db
```

Instalación de PostgreSQL

En la documentación oficial vamos a encontrar las instrucciones para instalar PostgreSQL en los diferentes sistemas operativos, como así también vamos a encontrar diferentes versiones. En ese sentido, para las prácticas que vamos a realizar, se sugiere instalar a partir de la versión 12 en adelante.

- Documentación oficial: <https://www.postgresql.org/>

Pasos generales para trabajar con PostgreSQL

1. **Crear una conexión a la base de datos:** utilizando el método `connect()` generamos un objeto de conexión a base de datos SQL Server.
2. **Crear un cursor:** El método `cursor()` se utiliza para realizar la conexión y ejecutar consultas SQL que nos permiten crear tablas, insertar datos, etc. Para crear un cursor solo necesitamos usar la conexión que ya hemos creado.
3. **Ejecutar una sentencia SQL:** Una vez creado el cursor, podremos ejecutar las sentencias SQL utilizando el método `execute()`.
4. **Realizar un commit:** El método `commit()` se utiliza para guardar cualquier transacción de forma permanente en el sistema de base de datos. Todas las modificaciones de datos o del



Tema 2. Tratamiento de datos con Pandas


Objetivo

Una vez finalizado este tema estarás capacitado para las siguientes acciones:

- Conocer y utilizar algunas de las funciones principales de Pandas para la visualización y tratamiento de data frames.
- Crear subconjuntos utilizando los métodos loc e iloc
- Crear subconjuntos a partir de condiciones lógicas
- Trabajar con copias de data frames.
- Utilizar el parámetro inplace para modificar un dataframe.









Notebook práctico

En el siguiente [notebook](#) vamos a trabajar con algunos de los métodos más utilizados de la librería Pandas y luego vamos a enfocarnos en diferentes técnicas de filtrado y tratamiento de data frames, que nos será de mucha utilidad al momento de realizar un análisis exploratorio de datos.


jupyter

Datos con Pandas II
Last Checkpoint: hace 2 horas (autosaved)
Logout

File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)









Markdown

Datos con Pandas II

Utilizar la librería Pandas para filtrar datos. Comprender conceptos de copias de dataframes utilizando el parámetro inplace

In [1]: `# Importamos la librería Pandas. En caso de no estar instalada, ejecutar --> pip install pandas`
`import pandas as pd`

Dataset a utilizar: Winter olympic medals

Este dataset se encuentra disponible en la web y nos brinda información de los ganadores de medallas olímpicas entre 1924 y 2006

In [2]: `url = 'http://winterolympicsmedals.com/medals.csv'`
`df = pd.read_csv(url)`
`df`

Out[2]:

	Year	City	Sport	Discipline	NOC	Event	Event gender	Medal
0	1924	Chamonix	Skating	Figure skating	AUT	individual	M	Silver
1	1924	Chamonix	Skating	Figure skating	AUT	individual	W	Gold
2	1924	Chamonix	Skating	Figure skating	AUT	pairs	X	Gold
3	1924	Chamonix	Bobsleigh	Bobsleigh	BEL	four-man	M	Bronze
4	1924	Chamonix	Ice Hockey	Ice Hockey	CAN	ice hockey	M	Gold
...
2306	2006	Turin	Skiing	Snowboard	USA	Half-pipe	M	Silver
2307	2006	Turin	Skiing	Snowboard	USA	Half-pipe	W	Gold
2308	2006	Turin	Skiing	Snowboard	USA	Half-pipe	W	Silver
2309	2006	Turin	Skiing	Snowboard	USA	Snowboard Cross	M	Gold
2310	2006	Turin	Skiing	Snowboard	USA	Snowboard Cross	W	Silver

2311 rows × 8 columns



Cierre

Durante esta unidad aprendimos a conectarnos a diferentes RDBMS y a trabajar con Pandas.

Comenzamos viendo los diferentes adaptadores que nos permiten conectarnos a RDBMS SQLite, MySQL, PostgreSQL y SQL Server.

Una vez conectados, aprendimos a ejecutar diferentes sentencias SQL que nos permitieron crear una base de datos, una tabla, insertar registros y consultarlas, entre otros.

Luego utilizamos SQLAlchemy para conectarnos a los diferentes RDBMS, lo cual nos permitió utilizar Pandas para consultar tablas y guardar los datos como dataframes, y también escribir registros en la base de datos a partir de un dataframe.

Finalmente aprendimos a extraer subconjuntos de datos en dataframes, utilizando diferentes estrategias que nos permitieron extraer tanto filas como columnas.

Quedamos a disposición de las consultas que puedan surgir.



Referencias

<https://docs.sqlalchemy.org/en/14/core/engines.html>

<https://www.sqlite.org/index.html>

<https://www.postgresql.org/docs/>

<https://dev.mysql.com/doc/refman/8.0/en>

<https://www.microsoft.com/es-es/sql-server/>

tiiiiiit by 