

**Unidad 3. MACHINE LEARNING PARA LA CIENCIA DE
DATOS**

SQL avanzado y otros datos

Consultas de mayor complejidad

WHERE

La cláusula WHERE se utiliza para filtrar registros.
También se usa para extraer solo aquellos registros que cumplen una condición específica.

AND, OR, NOT

La cláusula WHERE se puede combinar con los operadores AND, OR y NOT.

1. El operador AND muestra un registro si todas las condiciones separadas por AND son VERDADERAS.
2. El operador OR muestra un registro si alguna de las condiciones separadas por OR es VERDADERA.
3. El operador NOT muestra un registro si la(s) condición(es) NO ES VERDADERA.

SELECT TOP

- La cláusula SELECT TOP se usa para especificar el número de registros a devolver.
- Es útil en tablas grandes con miles de registros si no se quiere devolver toda la cantidad de registros después del filtro

ALIASES

Los alias de SQL se utilizan para dar a una tabla, o a una columna de una tabla, un nombre temporal.

Se utilizan a menudo para hacer que los nombres de las columnas sean más legibles. Un alias solo existe mientras dure esa consulta.

Se crea un alias con la palabra clave AS.

COMMENTS

Los comentarios se utilizan para explicar secciones de instrucciones SQL o para evitar la ejecución de instrucciones SQL.

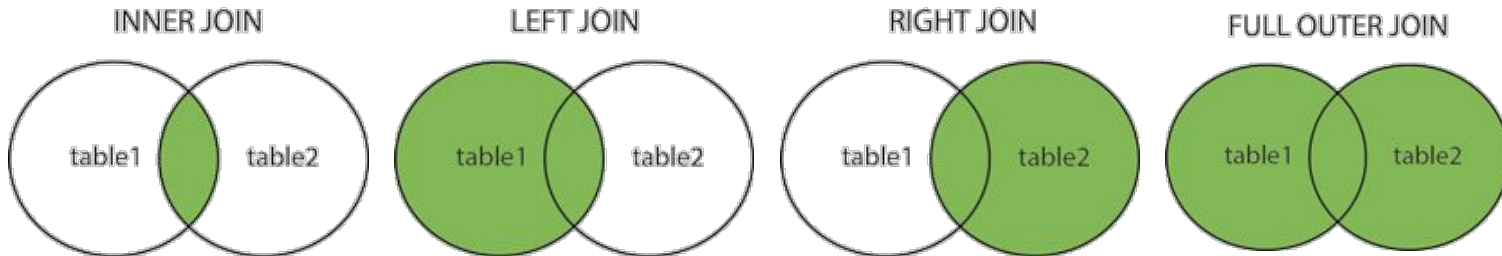
1. Los comentarios de una sola línea comienzan con `--`. Cualquier texto entre `--` y el final de la línea será ignorado (no será ejecutado).
2. Los comentarios de varias líneas comienzan con `/*` y terminan con `*/`. Cualquier texto entre `/*` y `*/` será ignorado.

Joins, Agregaciones y Agrupaciones

JOINS

JOINS

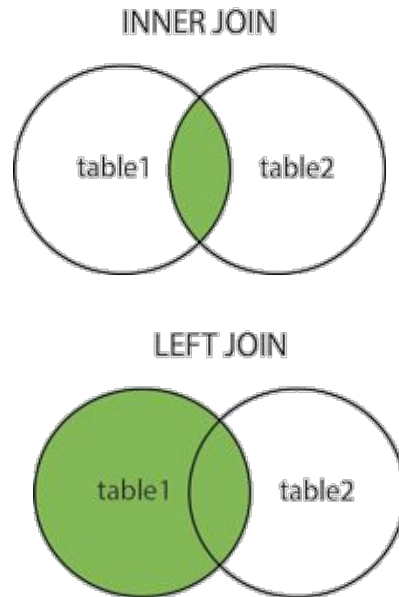
Una cláusula JOIN se usa para combinar filas de dos o más tablas, en función de una columna relacionada entre ellas. Existen 4 tipos comunes (INNER, LEFT, RIGHT, FULL)



INNER, LEFT JOIN

INNER JOIN selecciona registros que tienen valores coincidentes en ambas tablas.

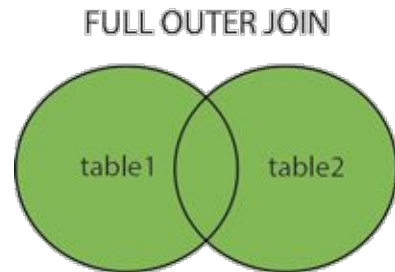
LEFT JOIN devuelve todos los registros de la tabla de la izquierda (tabla1) y los registros coincidentes de la tabla de la derecha (tabla2).



RIGHT, FULL JOIN

RIGHT devuelve todos los registros de la tabla de la derecha (tabla2) y los registros coincidentes de la tabla de la izquierda (tabla1).

FULL OUTER JOIN devuelve todos los registros cuando hay una coincidencia en los registros de la tabla izquierda (tabla1) o derecha (tabla2).



GROUP BY

1. Agrupa filas que tienen los mismos valores en filas de resumen, como "encontrar la cantidad de clientes en cada país".
2. Se usa a menudo con funciones agregadas (COUNT(), MAX(), MIN(), SUM(), AVG()) para agrupar el conjunto de resultados por una o más columnas.

HAVING

1. La cláusula HAVING se agregó a SQL porque la palabra clave WHERE no se puede usar con funciones agregadas.
2. Se usa en combinación con la cláusula GROUP BY para restringir los grupos de filas devueltas a solo aquellas cuya condición es VERDADERA.

MIN, MAX

MIN, MAX

- La función MIN() devuelve el valor más pequeño de la columna seleccionada.
- La función MAX() devuelve el valor más grande de la columna seleccionada.

COUNT, AVG, SUM

1. La función COUNT() devuelve el número de filas que coinciden con un criterio específico.
2. La función AVG() devuelve el valor promedio de una columna numérica.
3. La función SUM() devuelve la suma total de una columna numérica.



SQL con JOINS

Resolveremos un problema real utilizando Joins y funciones de agrupación

Duración: **15 mins**



ACTIVIDAD EN CLASE

SQL con JOINS

Utilizaremos la base de datos en [este enlace](#)

1. Selecciona todos los pedidos con información del cliente y del remitente (Pueden hacer uso de las tablas Orders y Customers, la columnas que permite el JOIN son CustomerID y ShipperID)
2. Obtener el número de pedidos enviados por cada remitente (Pueden hacer uso de las tablas Shippers y Orders, el JOIN se realiza por medio de la columna ShipperID)

1. Selecciona todos los pedidos con información del cliente y del remitente (Pueden hacer uso de las tablas Orders y Customer's, la columnas que permite el JOIN son CustomerID y ShipperID)

```
SELECT  
Orders.OrderID,  
Orders.OrderDate,  
Customers.ContactName AS Cliente,  
Shippers.ShipperName AS Remitente  
FROM ((Orders  
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID)  
INNER JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID);
```

OrderID	OrderDate	Cliente	Remitente
10290	8/27/1996	Pedro Afonso	Speedy Express
10284	8/19/1996	Renate Messner	Speedy Express
10388	12/19/1996	Hari Kumar	Speedy Express
10390	12/23/1996	Roland Mendel	Speedy Express
10296	9/3/1996	Carlos González	Speedy Express
10349	11/8/1996	Art Braunschweiger	Speedy Express
10309	9/19/1996	Patricia McKenna	Speedy Express
10401	1/1/1997	Paula Wilson	Speedy Express

2. Obtener el número de pedidos enviados por cada remitente (Pueden hacer uso de las tablas Shippers y Orders, el JOIN se realiza por medio de la columna ShipperID)

```
SELECT  
    Shippers.ShipperName,  
    COUNT(Orders.OrderID) AS NumeroDePedidos  
FROM Orders  
INNER JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID  
GROUP BY Shippers.ShipperName;
```

ShipperName	NumeroDePedidos
Federal Shipping	68
Speedy Express	54
United Package	74

Conexión a modelos relacionales SQL usando Pandas

Cómo Acceder a Bases relacionales desde Pandas

Librerías para conectar SQL-Pandas

Cargar datos a un DataFrame puede ser bastante directo con las siguientes librerías:

1. Sqlalchemy
2. Sqlite3
3. cx_Oracle
4. Pyodbc
5. etc.

Sqlite3

Sqlite3

Con Pandas, es muy sencillo combinar el motor particular que se use y transformarlo a un DataFrame usando sql como string.

Creamos la conexión con sqlite3 y junto con una query sql, la usamos de argumento con el método **read_sql_query** de pandas.

```
con = sqlite3.connect(base_sql)
df = pd.read_sql_query(sql, con)
```

Sqlite3

1. Creamos la query sql como str para extraer todas las tablas de la base.
2. Generamos la conexión hacia la base **nba_salary.sqlite**.
3. Integramos el dataframe con la query y la conexión creada con anterioridad.
4. Se hace un print del DataFrame (En este caso solo tiene dos elementos) y se cierra la conexión.

```
1 import pandas as pd
2 import sqlite3
3
4 sql = "SELECT name FROM sqlite_master WHERE type='table'"
5
6
7 # Read sqlite query results into a pandas DataFrame
8 con = sqlite3.connect("nba_salary.sqlite")
9 df = pd.read_sql_query(sql, con)
10
11 # Verify that result of SQL query is stored in the dataframe
12 print(df)
13
14 con.close()
```

```
              name
0  NBA_season1718_salary
1           Seasons_Stats
```

Sqlite3

Mediante query, también podemos extraer la tabla **NBA_season1718_salary**:

```
1 import pandas as pd
2 import sqlite3
3
4 con = sqlite3.connect("nba_salary.sqlite")
5 df = pd.read_sql_query("SELECT * From NBA_season1718_salary", con)
6 df.head()
```

	X1	Player	Tm	season17_18
0	1.0	Stephen Curry	GSW	34682550.0
1	2.0	LeBron James	CLE	33285709.0
2	3.0	Paul Millsap	DEN	31269231.0
3	4.0	Gordon Hayward	BOS	29727900.0
4	5.0	Blake Griffin	DET	29512900.0

Sqlalchemy

Sqlalchemy

Sqlalchemy puede leer cualquier tipo de base. Donde dice sqlite se puede cambiar el motor:

```
import sqlalchemy as db
engine = db.create_engine('sqlite:///nba_salary.sqlite')
connection = engine.connect()
```

DataFrame

```
import pandas as pd
df = pd.read_sql_query("SELECT * From Seasons_Stats", connection)
```

Introducción a Github

Git

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente



GitHub

GitHub es una plataforma de control de versiones distribuida donde los usuarios pueden colaborar o adoptar proyectos de código fuente abierto, fork code, compartir ideas y más.

Su mayor contribuidor es Microsoft y ofrece un servicio de Software as Service (SaaS) creado en 2008



Diferencias Git vs. Github

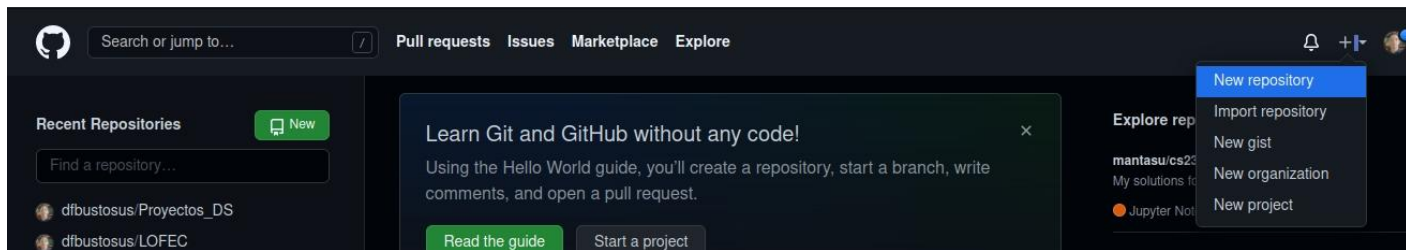
Git	Github
Instalado localmente	Hosteado en la nube
Lanzado en 2005 y mantenido por The Linux Foundation	Lanzado en 2008 y mantenido por Microsoft
Se enfoca en control de versiones y código compartido	Se enfoca en un hosting de código centralizado
Se ejecuta en la terminal	Administrado en la web
Provee una interface desktop llamada Git Gui	Provee una interfaz desktop llamada Github Desktop
Pocas configuraciones externas disponibles	Marketplace diverso para integración de herramientas
Licencia Open Source	Versión gratuita y también paga

Cómo usar Github

Cómo usar Github

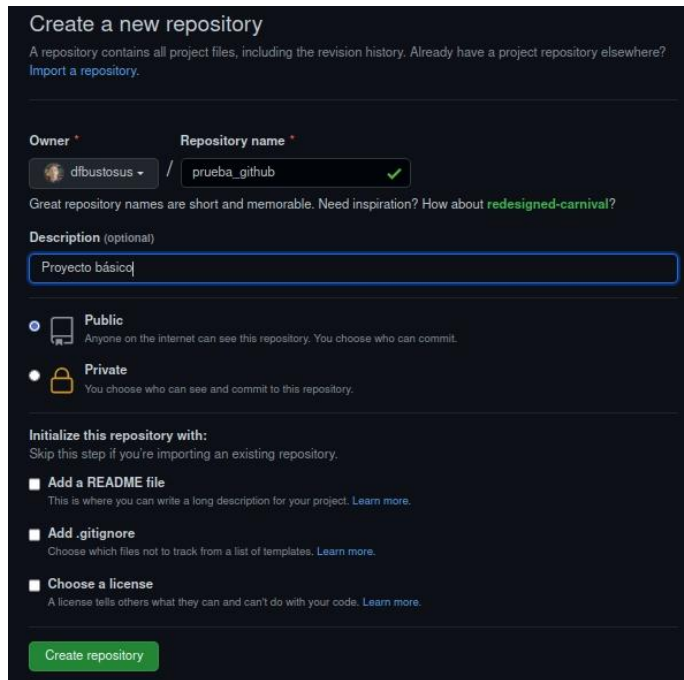
Si solo deseas realizar un seguimiento de tu código localmente, no necesitas usar GitHub. Pero si deseas trabajar con un equipo, puedes usar GitHub para modificar el código del proyecto de forma colaborativa.

1. Para crear un nuevo repositorio en GitHub, inicie sesión y vaya a la página de inicio de GitHub. Puede encontrar la opción "Nuevo repositorio" debajo del signo "+" junto a su foto de perfil, en la esquina superior derecha de la barra de navegación:



Cómo usar Github

- Después de hacer clic en el botón, GitHub le pedirá que asigne un nombre a su repositorio y proporcione una breve descripción.
- Cuando haya terminado de completar la información, presione el botón 'Crear repositorio' para crear su nuevo repositorio.

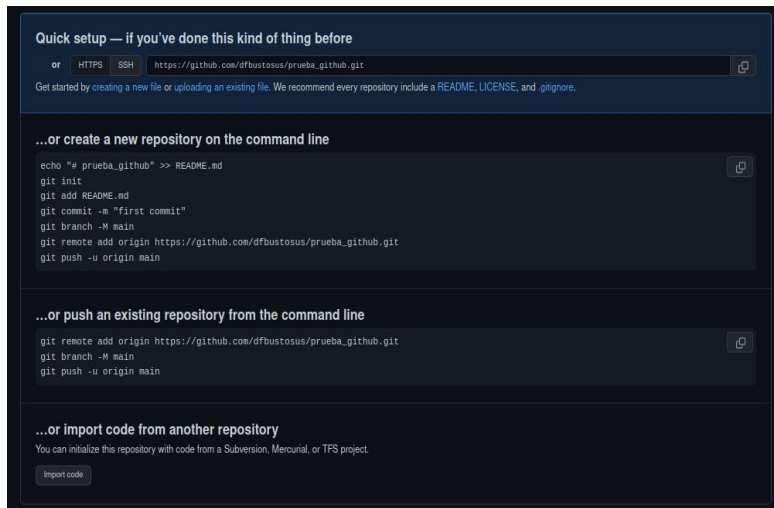


The screenshot shows the 'Create a new repository' page on GitHub. At the top, it says 'Create a new repository' and provides a brief explanation of what a repository is. Below this, there are two main sections: 'Owner' and 'Repository name'. The 'Owner' is set to 'dibustosus' and the 'Repository name' is 'prueba_github', which has a green checkmark next to it. A note below these fields says 'Great repository names are short and memorable. Need inspiration? How about redesigned-carnival?'. The 'Description (optional)' field contains the text 'Proyecto básico'. Below the description field, there are two radio buttons for visibility: 'Public' (selected) and 'Private'. The 'Public' option is described as 'Anyone on the internet can see this repository. You choose who can commit.' and the 'Private' option is described as 'You choose who can see and commit to this repository.' Below these options, there is a section titled 'Initialize this repository with:' which includes three checkboxes: 'Add a README file' (selected), 'Add .gitignore' (selected), and 'Choose a license' (selected). Each checkbox has a brief description and a 'Learn more' link. At the bottom of the form is a green button labeled 'Create repository'.

Cómo usar Github

4. GitHub le preguntará si desea crear un nuevo repositorio desde cero o si desea agregar un repositorio que haya creado localmente.

En este caso, dado que ya hemos creado un nuevo repositorio localmente, queremos enviarlo a GitHub, lo podemos hacer desde la terminal



JavaScript Object Notation

Javascript Object Notation (JSON)

- ✓ Se trata de un formato de datos **semi-estructurado no tabular**, es decir que los registros no tienen que tener un mismo conjunto de atributos.
- ✓ Se utiliza mucho para **transferir datos entre servidores y clientes**. La fuente más común donde se consumen los JSON son las interfaces de programación de aplicaciones web o **web APIs**.
- ✓ Los datos se organizan en colecciones de objetos o paquetes.
- ✓ Pueden **"anidarse"** → Los valores pueden ser listas u otros objetos.

Javascript Object Notation (JSON)

```
{
  "empid": "SJ011MS",
  "personal": {
    "name": "Smith Jones",
    "gender": "Male",
    "age": 28,
    "address": {
      "streetaddress": "7 24th Street",
      "city": "New York",
      "state": "NY",
      "postalcode": "10038"
    }
  },
  "profile": {
    "designation": "Deputy General",
    "department": "Finance"
  }
}
```

www.kodingmadesimple.com

Ejemplo de un JSON anidado

Se observa por ejemplo que puede contener diccionarios dentro de una key.

```
1 [
2   {
3     "id": 100,
4     "src": "7.18.1.4",
5     "dest": "1.2.3.4",
6     "src": "tcp/22",
7     "svrs": "chicago",
8     "desc": "Data Centers in Group A"
9   },
10  {
11    "id": 101,
12    "src": "7.18.1.4",
13    "dest": "1.2.3.4",
14    "src": "tcp/80",
15    "svrs": "chicago,new york",
16    "desc": "Data Centers in Group B"
17  }
18 ]
```

Librería JSON

<https://docs.python.org/3/library/json.html>

¡Python admite JSON de forma nativa! Python viene con un paquete integrado llamado json para codificar y decodificar datos JSON.

```
1 import json|
```



Hands on lab

Realizaremos dos actividades de clase para comprender y **llevar a la práctica** los conceptos teóricos vistos.

¿De qué manera?

En primera instancia, trabajaremos con visualizaciones de datos de manera individual. Luego, trabajamos colaborativamente compartiendo los avances del Proyecto Final.

Tiempo estimado: **35/40 minutos**

Título

Consigna: De manera individual, vamos a leer este ejemplo guardado en carpeta llamado “[ejemplo 1.json](#)”

Luego de leerlo, los y las invitamos a anotar ciertas impresiones que hayan tenido a la hora de leerlo.

¿Qué les pareció el ejemplo?

Entre todos, analicemos el caso.
Los/as invitamos a ir dialogando a través del chat.

Librería JSON

```
import json
```

```
1 file_json = open('ejemplo_1.json')  
2 data = json.load(file_json)  
3 data
```

```
{'Duration': {'0': 60, '1': 60, '2': 60, '3': 45, '4': 45, '5': 60},  
 'Pulse': {'0': 110, '1': 117, '2': 103, '3': 109, '4': 117, '5': 102},  
 'Maxpulse': {'0': 130, '1': 145, '2': 135, '3': 175, '4': 148, '5': 127},  
 'Calories': {'0': 409, '1': 479, '2': 340, '3': 282, '4': 406, '5': 300}}
```

Librería JSON con Pandas

Método: `read_json()`:

- ✓ Puede aplicarse como path a un archivo JSON o una url que termine .json o directamente un string.
- ✓ Pandas adivina el tipo de datos pero tb puede ser especificado (dtype)
- ✓ **Orientación:** Consiste en el parámetro que indica cómo se estructura JSON, para poderlo convertir en un DataFrame.

¿Preguntas?

Glosario

Git: software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente

Github: es una plataforma de control de versiones distribuida donde los usuarios pueden colaborar o adoptar proyectos de código fuente abierto, fork code, compartir ideas y más.

JSON: Se trata de un formato de datos **semi-estructurado no tabular**, es decir que los registros no tienen que tener un mismo conjunto de atributos.

API: Es una **interfaz que permite la comunicación entre dos sistemas distintos** permitiendo agregar diversas funciones a sitios web y aplicaciones.

sqlite y sqlalchemy: librerías para realizar conexiones con bases de datos relacionales con Pandas

CLASE N°4

Glosario

TCL: son comandos (COMMIT,, ROLLBACK) que permiten manejar transacciones en una base de datos relacional.

Consultas de mayor complejidad: aquellas donde se utiliza clausulas AND, OR, NOT, WHERE, SELECT TOP, LIKE, ALIASING, COMMENTS o CASE-END con el fin de obtener un resultado de utilidad según el caso

Transacción: secuencia de operaciones desarrolladas (con uno o más comandos SQL)

Otros tipos de comandos: incluyen por ejemplo cláusulas como IN, BETWEEN, ANY, ALL que permiten hacer filtros de tipo booleano a la hora de obtener resultados de una consulta SQL

JOINS: se utilizan para combinar los resultados de una tabla con otras teniendo en cuenta una columna en común

Agrupaciones y agregaciones: cláusulas que permiten agrupar por categorías y obtener medida de resumen de las variables deseadas