

Unidad N°1 Aplicación de herramientas informáticas al Ciclo de Vida de los Datos - Fase !

Sitio: [Centro de E-Learning - UTN.BA](#)

Curso: Curso de Data Science

Libro: Unidad N°1 Aplicación de herramientas informáticas al Ciclo de Vida de los Datos - Fase !

Imprimido por: Virginia Marich

Día: Wednesday, 27 de December de 2023, 09:52

Descripción

Tabla de contenidos

1. Introducción

2. Extracción de datos y el rol de SQL

- 2.1. ¿Qué tipos de SGBD existen?
- 2.2. Tipos de relaciones entre Bases de Datos Relacionales
- 2.3. ¿Qué es SQL (Structured Query Language)?
- 2.4. ¿Cómo realizar la descarga e instalación de SQL Server y Management Studio?
- 2.5. ¿Cuáles son los comandos básicos con los que puedes extraer información de una tabla de datos?
- 2.6. Ejemplos de sintaxis DQL para comprender el funcionamiento de cada comando y cláusula
- 2.7. Ejemplos de sintaxis DML para comprender el funcionamiento de cada comando y cláusula
- 2.8. Funciones de agregación en SQL
- 2.9. Funciones escalares
- 2.10. Subconsultas
- 2.11. ¿Qué es un JOIN?
- 2.12. Automatizaciones dentro de SQL

3. Exploración y Visualización de datos y el rol de las herramientas de visualización más valoradas en la industria: Tableau y Power BI

- 3.1. Acciones prácticas en Power BI y Tableau que se aprovechan desde el rol del data scientist.

4. Conclusiones

5. Bibliografía utilizada y sugerida

1. Introducción

¡Bienvenidos a este emocionante capítulo donde vamos a explorar cómo las herramientas informáticas nos permiten trabajar con los datos en todo su ciclo de vida!

En este capítulo, vamos a adentrarnos en el uso de herramientas clave de Data Science como SQL, Power BI y Tableau para adquirir, explorar y visualizar los datos. Aprenderemos cómo estas herramientas se utilizan para dar sentido a los datos y transformarlos en información útil para tomar decisiones informadas y estratégicas.

Imaginen poder extraer información de bases de datos, crear visualizaciones atractivas y dinámicas. ¡Estas habilidades son esenciales para cualquier estudiante de Data Science que quiera destacarse en la industria!

Al final de este capítulo, estarán listos para aplicar estas habilidades en situaciones reales y llevar su carrera en Data Science al siguiente nivel. ¡Así que prepárense para un emocionante viaje en el uso de herramientas informáticas aplicadas al ciclo de vida de los datos!

2. Extracción de datos y el rol de SQL

¿Qué es una base de datos relacional?

Bases de datos

Retomemos nuevamente el tema de las bases de datos, como definición es un conjunto de datos almacenado en un formato específico, qué, interrelacionados por un contexto en común lo convierten en información.

Tipos de Bases de Datos:

Relacional

Bases de datos relacional es un sistema en el que un conjunto de tablas están relacionadas entre sí, a través de campos específicos: clave primaria y clave secundaria o foránea.

Recordemos que una tabla está conformada por la siguiente tupla:

1. Filas/registros
2. Columnas/atributos

ID ESTUDIANTES	NOMBRE	APELLIDO	FECHA DE INGRESO	CARRERA	FILA
01	Raúl	Fonseca	Marzo 2004	Ingeniería industrial	
02	Ezequiel	Ramirez	Marzo 2010	Abogacia	
03	Sofia	Benegas	Marzo 2013	Medicina	

COLUMNA

No relacionales

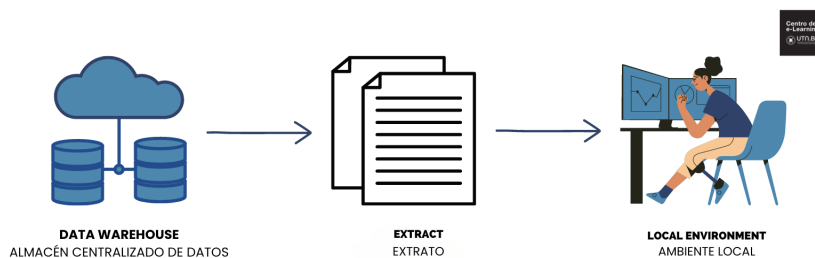
No tienen un identificador que sirva de relación entre un conjunto de datos y otros. La información no se presenta como la conocemos nosotros, son datos no estructurados, aleatorios que se van guardando o la información se presenta mediante objetos como en la programación orientada a objetos

La mayoría de las organizaciones que requieren data analyst utilizan bases de datos relacionales pero la mayoría de las que requieren data scientist requieren ambas: relacionales y no relacionales, por su gran volumen de datos.

¿Cómo se comunican los Data Scientist con estas bases de datos?

Sistema de Gestión de Base de Datos

Vamos a empezar este tema recordando el diagrama visto en la Unidad 1 sobre extracción de los datos:



Recordemos que podemos tener dos instancias de bases de datos en una empresa:

- Entorno de producción: son los datos en tiempo real con los que está trabajando la empresa y cualquier modificación puede ser crítica.
- Entorno de testing: donde por lo general se trabaja desde este rol y casi todos en el ambiente IT para realizar análisis, pruebas y extracciones a entornos locales.

Aquí es donde ingresan los sistemas de gestión de bases de datos. Un sistema de gestión de base de datos (SGBD o DBMS, por sus siglas en inglés) proporcionan una interfaz para que los usuarios creen y gestionen bases de datos, y ofrecen una serie de herramientas y servicios para que los usuarios accedan a la información almacenada en las bases de datos.

Un buen SGBD se caracteriza por los siguientes atributos:

Recuperación	Concurrencia	Integridad	Seguridad
Capacidad de proteger los datos ante fallos en el sistema o en las aplicaciones	Permiten que muchas transacciones puedan acceder a una misma base de datos a la vez	Monitorea que la base de datos mantenga una congruencia con cada campo agregado	Garantiza la seguridad a toda la información almacenada mediante el sistema encargado de administrar la privacidad

Recuperación - backup

Concurrencia - muchos al mismo tiempo:

- Usuarios primarios (SAP, Tango, que necesitan bajar información)
- Usuarios especializados (que acceden directamente para hacer consultas)
- Usuarios desarrolladores (se conectan para usarlo para algún aplicativo)
- Usuarios que mantienen y construyen la estructura además de asignar permisos (DB users)

2.1. ¿Qué tipos de SGBD existen?

En general, la elección del SGBD dependerá del tipo de datos que se estén manejando y de las necesidades específicas del proyecto, entre los más comunes:

- SGBD relacionales: los datos se almacenan en tablas relacionales y se pueden acceder mediante el lenguaje SQL (Structured Query Language).
- SGBD no relacionales: se utilizan para almacenar y recuperar datos en formas diferentes a las tablas relacionales, como documentos, gráficos o datos de series temporales.

SQL es un lenguaje fundamental para trabajar con bases de datos relacionales, y su demanda en el mercado laboral lo convierte en un conocimiento imprescindible para cualquier profesional que trabaje con datos.

Aunque los sistemas de gestión de bases de datos no relacionales son cada vez más populares en la industria del Análisis de Datos, su aprendizaje puede resultar más complejo que el de SQL y exige un enfoque más exhaustivo el cual excede la complejidad de este capítulo. Pero si buscas profundizar en los sistemas de gestión de bases de datos no relacionales, te recomiendo explorar recursos especializados una vez finalizado este curso. De esta manera, podrás continuar tu camino de aprendizaje en el fascinante mundo de la gestión de bases de datos:


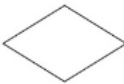
1. ["What is NOSQL?"](#) Este artículo de la documentación oficial de MongoDB proporciona una introducción clara y fácil de entender a los sistemas de gestión de bases de datos NoSQL.
2. [What is Amazon DynamoDB?](#) Este artículo de la documentación oficial de DynamoDB proporciona una visión general de los modelos de datos NoSQL y cómo diseñarlos.

Justificado el por qué abordaremos sólo el conocimiento en bases de datos relacionales en este curso, continuamos con la explicación de los diagramas más famosos para explicar visualmente la relación entre bases de datos:

Diagrama de Entidad - Relación

El diagrama de entidad-relación es el plano de una base de datos que sirve como guía para su construcción y mantenimiento. Al igual que un arquitecto se guía por los planos para construir una casa, un administrador de bases de datos utiliza el diagrama de entidad-relación para crear y mantener la base de datos.

El proceso de construcción comienza con la elaboración del diagrama entidad-relación, en el que cada parte del esquema se dibuja en función de lo que representa: rectángulos para tablas o entidades, elipses para columnas o atributos/campos, rectángulos para relaciones entre tablas que representan acciones, y líneas que unen acciones y entidades.

SÍMBOLO	NOMBRE	SIGNIFICADO
	Rectángulo	Conjunto entidad
	Óvalo	Atributo
	Diamante	Relación
	Línea	Ligas: atributo a entidad

Si bien es cierto que la construcción del diagrama de entidad-relación es tarea del DBA, comprender su estructura y significado es una ventaja para el Data Scientist al momento de realizar consultas y extraer información de la base de datos.

Un buen diagrama de entidad-relación, construido de manera apropiada, evita la duplicidad de los datos en una misma tabla y permite que cada tabla incorpore información nueva y se relacione fácilmente con otras tablas, lo que se conoce como normalización y dependencia funcional entre tablas. Al entender estos conceptos, el Data Scientist podrá diseñar consultas más efectivas y eficientes para extraer la información necesaria de la base de datos.

A continuación se ejemplifica el registro de las compras realizadas en el comedor universitario de una universidad realizado por un administrador del comedor, el cual se encargaba del mantenimiento de las transacciones.

En una primer medida los administradores de estos datos construyeron la siguiente tabla:

Index	Fecha	DNI	Nombre	Telefono	Importe
1	12/01/2020	37490011	Virginia Marich	15959551	2000
2	12/01/2020	33976995	Ezequiel Benegas	15241244	3000
3	12/01/2020	14132702	Roberto Uribe	15939214	2200
4	13/01/2020	37490011	Virginia Marich	15492911	1500
5	14/01/2020	14132702	Roberto Uribe	15241244	700

Reconocieron que tenían información duplicada de los estudiantes por lo que realizaron la modificación a dos tablas sin duplicidad:

ESTUDIANTES

DNI	Nombre	Telefono
37490011	Virginia Marich	15959551
33976995	Ezequiel Benegas	15241244
14132702	Roberto Uribe	15241244

COMPRAS EN EL COMEDOR

Indice de compra	DNI Cliente	Fecha	Importe
1	37490011	12/01/2020	2000
2	33976995	12/01/2020	3000
3	14132702	12/01/2020	2200
4	37490011	13/01/2020	1500
5	14132702	14/01/2020	700

Ahora sí podemos ahondar en el concepto de claves primarias y secundarias que mencionamos al explicar el concepto de bases de datos relacionales:

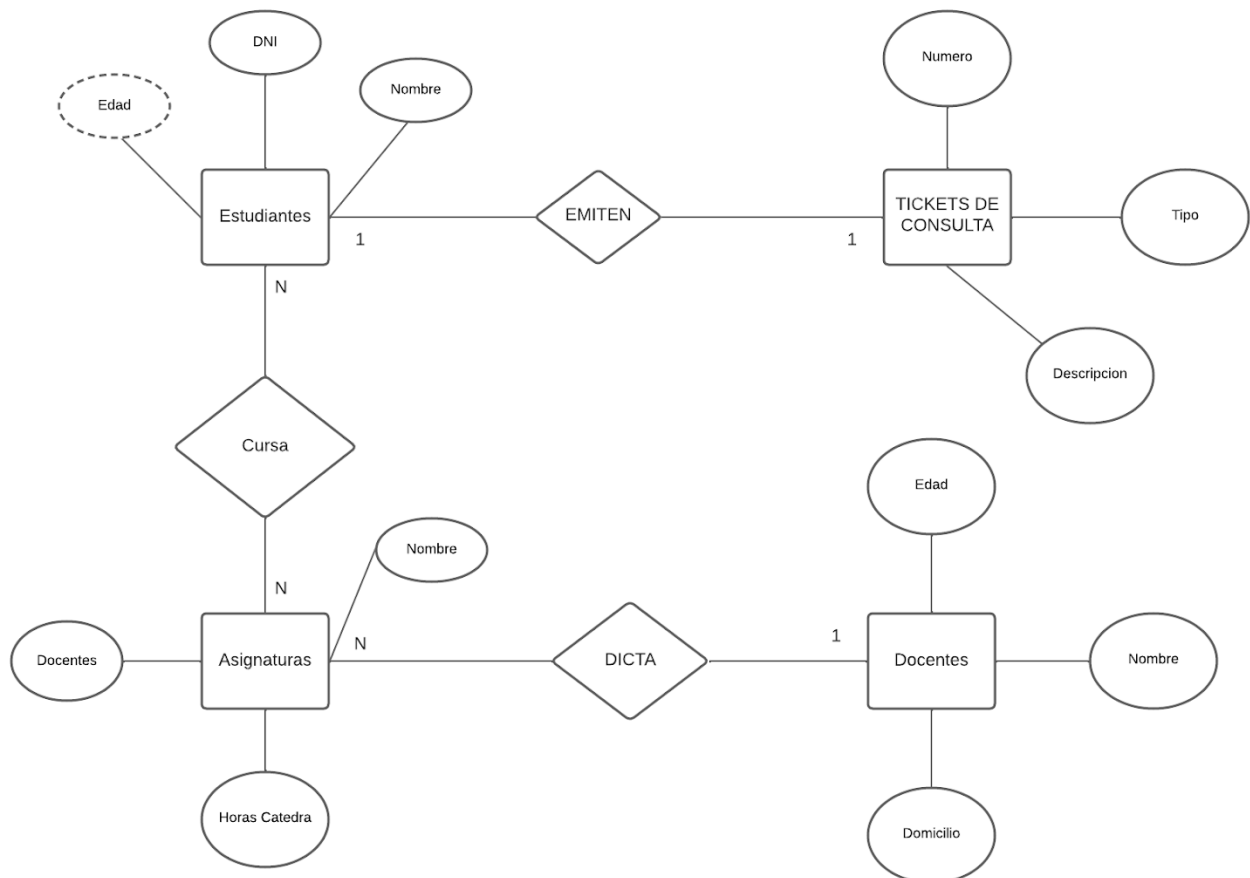
- Clave primaria o Primary Key: identificador único de registros de cada tabla. En la tabla estudiantes el Primary Key es el DNI del estudiante.
- Clave secundaria o Foreign Key: columna que permite relacionar tablas. Para la tabla de Compras el DNI del estudiante es la Foreign Key que es por medio de la cual se relaciona a la tabla Estudiantes.
- Muchas veces cuando nos enfrentamos a tablas también nos vamos a encontrar con claves llamadas Índice o Index que por default las considera el sistema como primary key, esta se puede reemplazar o utilizar como clave primaria dependiendo la lógica de negocio o análisis.

2.2. Tipos de relaciones entre Bases de Datos Relacionales

A esto en la jerga se le denomina "cardinalidad" y se refiere a la relación entre dos entidades en una base de datos.

Existen tres tipos comunes de cardinalidad en bases de datos:

1. Cardinalidad uno-a-uno (1:1): En este caso, una instancia de una entidad está relacionada con una sola instancia de otra entidad y viceversa. Por ejemplo, un estudiante puede tener solo una identificación de estudiante y una identificación de estudiante se puede asignar a un solo estudiante.
2. Cardinalidad uno-a-muchos (1:N): En este caso, una instancia de una entidad está relacionada con varias instancias de otra entidad, pero cada instancia de la segunda entidad sólo está relacionada con una instancia de la primera entidad. Por ejemplo, un estudiante puede haber generado varios tickets de consultas, pero cada ticket único solo está relacionado con un solo estudiante.
3. Cardinalidad muchos-a-muchos (N:M): se refiere a una relación donde muchas instancias de una entidad están relacionadas con muchas instancias de otra entidad. Para implementar esta relación, se crea una tabla intermedia que asocia las dos tablas principales. Por ejemplo, un estudiante puede estar inscrito en muchos cursos y un curso puede tener muchos estudiantes inscritos.



Aquí hay un recurso de un caso real de cómo se han creado diagramas de entidad relación en la estructura organizacional de spotify para organizarse de manera ágil

Spotify Engineering Culture (by Henrik Kniberg)"

Disponible en <https://www.youtube.com/watch?v=Yvfz4HGtoPc>

2.3. ¿Qué es SQL (Structured Query Language)?

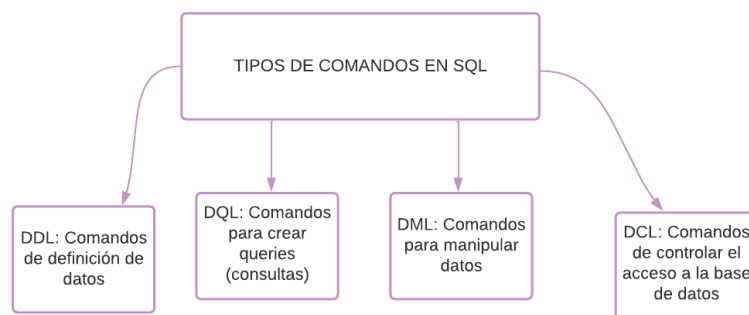
Teniendo en cuenta la introducción sobre las bases de datos relacionales y el hecho de que el lenguaje adecuado para acceder a ellas es SQL, podemos profundizar en la herramienta informática que se utiliza para manejar esta tecnología.

SQL, qué significa Structured Query Language en inglés, es un lenguaje de consultas estructurado y un sistema de gestión de bases de datos que proporciona un lenguaje para expresar consultas y realizar modificaciones en la base de datos.

¿Qué podemos hacer en SQL?

1. Crear nuevas bases de datos
2. Crear nuevas tablas
3. Crear procedimientos
4. Crear vistas
5. Insertar, actualizar y eliminar registros
6. Ejecutar consultas
7. Recuperar datos
8. Establecer permisos en tablas, procedimientos y vistas.

Estas acciones pueden clasificarse dentro las siguientes categorías, las cuales engloban diferentes comandos que veremos más adelante, para ejecutar justamente las acciones antes mencionadas:



En SQL, los comandos se clasifican generalmente en cuatro categorías principales:

1. DDL (Data Definition Language): se utilizan para definir la estructura de la base de datos, como crear, modificar o eliminar tablas, vistas, índices y restricciones.
2. DML (Data Manipulation Language): se utilizan para manipular los datos en una base de datos, como insertar, actualizar y eliminar filas en una tabla.
3. DQL (Data Query Language): se utilizan para recuperar datos de una base de datos, como consultar y seleccionar datos de una tabla.
4. DCL (Data Control Language): se utilizan para controlar el acceso a la base de datos, como otorgar o revocar permisos de acceso a usuarios y roles.

Además, también hay otros comandos y categorías adicionales que se utilizan en SQL, como TCL (Transaction Control Language) para controlar las transacciones en una base de datos y SCL (Session Control Language) para controlar las sesiones de usuario.

Una analogía comúnmente utilizada para describir la manipulación de datos en SQL es la de una casa. En esta analogía, la base de datos se compara con una casa y los datos en la base de datos se comparan con los muebles y objetos en la casa.

En esta analogía, el sublenguaje DDL (Data Definition Language) se puede comparar con la construcción de la casa, incluyendo la creación de habitaciones, la instalación de ventanas, la construcción de paredes, etc. Es decir, DDL se utiliza para definir la estructura y el esquema de la base de datos.

El sublenguaje DML (Data Manipulation Language) se puede comparar con el proceso de mover y manipular objetos dentro de la casa. Por ejemplo, DML se utiliza para agregar, actualizar o eliminar objetos de la base de datos, como agregar nuevos datos, actualizar registros existentes o eliminar información antigua.

El sublenguaje DQL (Data Query Language) se puede comparar con buscar un objeto específico en una casa. Al igual que buscar una llave en un mueble específico, DQL se utiliza para hacer consultas en la base de datos y recuperar información específica de ella.

Por último, el sublenguaje DCL (Data Control Language) se puede comparar con la seguridad y el control de acceso a una casa: quién puede entrar, quien tiene la llave, quien puede entrar hasta qué lugar de la casa. Es decir, DCL se utiliza para establecer permisos y restricciones de acceso a la base de datos para diferentes usuarios o roles.

¿Por qué utilizar SQL en Ciencia de Datos?

Es importante para un data scientist tener habilidades en SQL porque muchas empresas y organizaciones utilizan bases de datos relacionales para almacenar y administrar grandes cantidades de datos. Al tener habilidades en SQL, un data scientist puede acceder, manipular y analizar estos datos de manera eficiente, lo que puede mejorar su productividad y eficacia en el trabajo. Además, el conocimiento de SQL también puede ser útil para trabajar con herramientas y tecnologías de análisis de datos, como Python, R y Tableau, que a menudo se integran con bases de datos relacionales.

Dicho esto, podemos entender que un data scientist utiliza principalmente comandos de DQL (Data Query Language) y DML (Data Manipulation Language) en su día a día, ya que su trabajo se centra en analizar y manipular datos en una base de datos para extraer información valiosa.

2.4. ¿Cómo realizar la descarga e instalación de SQL Server y Management Studio?

Descarga e Instalación

Conociendo la interfaz de SQL

En el lenguaje de SQL la forma de comunicación con las bases de datos se denomina “sentencias”, todas estas sentencias están en el idioma INGLÉS, son palabras reservadas para comunicarnos podríamos decir que son instrucciones, ¡Pero no te asustes! son bastantes sencillas.

Veamos algunas aclaraciones de la sintaxis:

- SQL no distingue de mayúsculas y minúsculas, por lo que estas sentencias/instrucciones pueden ser escritas de cualquier manera. Se recomienda como buena práctica hacerlo en MAYÚSCULAS.
- Al redactar las sentencias o instrucciones veremos que adquieren un color diferente en la interfaz.
- Se utilizan dos símbolos * y ; como buenas prácticas * = “todo” y ; = “fin”
- Los campos o nombres de columnas no deben tener “separaciones” o “espacios en blanco”, si tienen separaciones la forma de invocarlo es entre [].
Ya veremos mas adelante que es el “SELECT” pero para que no lo olvides te dejo un ejemplo

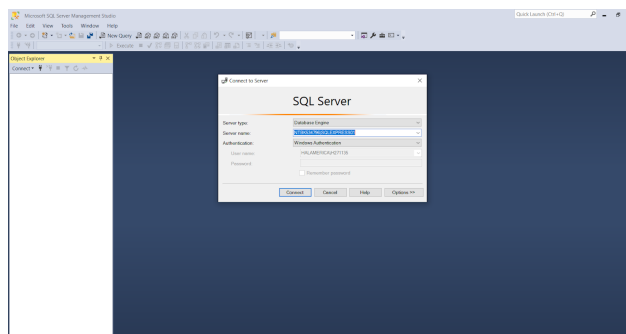
```
SELECT COUNT([Personajes Disney])
```

Si no existiera el ejemplo:

```
SELECT COUNT(PersonajesDisney)
```

Recurso para ampliar buenas prácticas: [5 Best Practices for writing SQL queries \(sqlshack.com\)](https://www.sqlshack.com/5-best-practices-for-writing-sql-queries/)

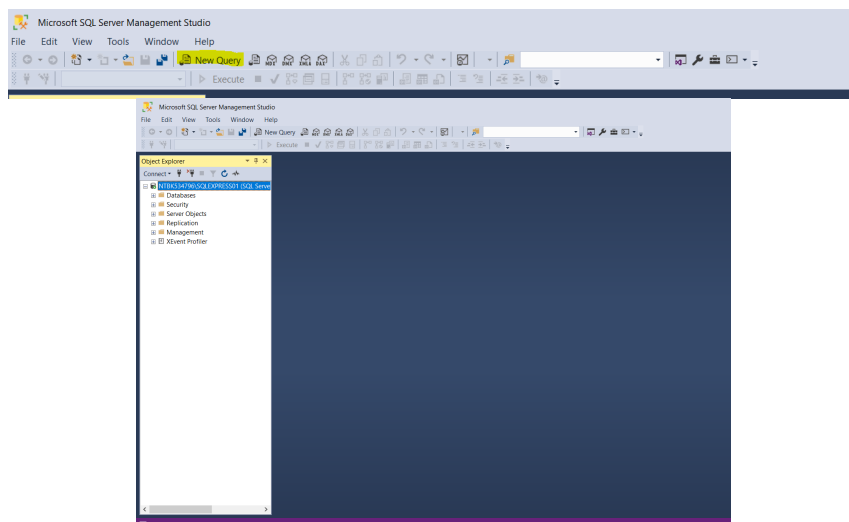
Ingresa y conocer la interfaz:



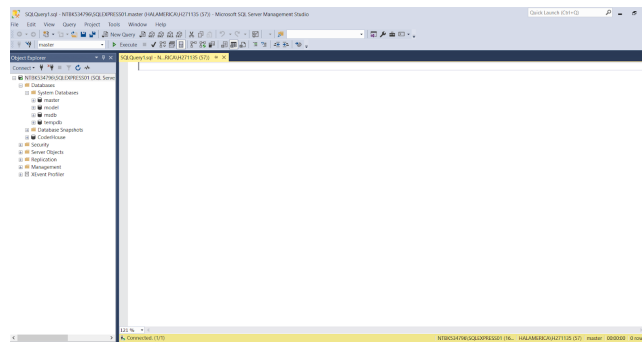
Ver la barra de exploración y despliegue de mi conexión de bases de datos. En Databases encuentro las bases de datos con las que trabajaré.

Acciones:

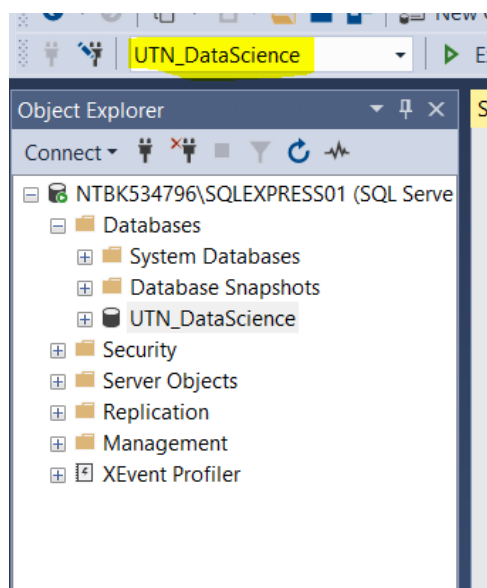
- New Query: para escribir un nuevo código o script.



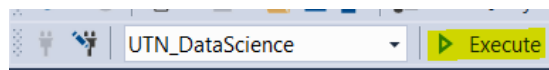
- Escribir lenguaje en Nuevo entorno de consulta luego de hacer click en New Query:



- Al realizar consultas siempre verificar que este ubicado en la base de datos con la que quiero trabajar en la parte superior izquierda del sistema se elige la misma.



- Luego de creada una sentencia, se resaltan las mismas y se selecciona EXECUTE para lograr que SQL ejecute la instrucción:



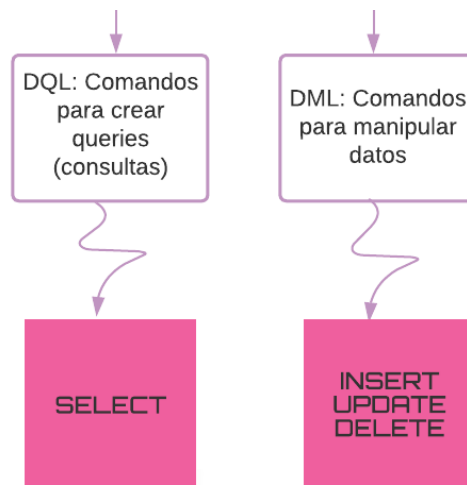
- Por default las tablas dentro de SQL se crean con la estructura `dbo.NombreTabla`
- Como Data Scientist podemos tener acceso a las tablas o vistas (tablas restringidas por seguridad con ciertos campos determinados), estas últimas también se crean con la estructura `dbo.NombreVista`
- Estos `dbo` son denominados “esquemas” dentro de la jerga, pero esa creación o modificación es parte del rol de los Data Base Administrator. Los esquemas son el compendio de tablas que puede acceder un usuario.

2.5. ¿Cuáles son los comandos básicos con los que puedes extraer información de una tabla de datos?

- En este curso sólo haré referencia a las sentencias utilizadas para hacer consultas que debería conocer un Data Scientist, pero si quieres conocer un listado más abarcativo para crear una base de datos, tablas e insertar datos te recomiendo el siguiente recurso [SQL Keywords Reference \(w3schools.com\)](https://www.w3schools.com/sql/sql_keywords.asp)
- Para repasar los tipos de datos y cómo se categorizan dentro de SQL, en el siguiente recurso [Tipos de datos SQL para MySQL, SQL Server y MS Access \(tecnologias-informacion.com\)](https://tecnologias-informacion.com/Tipos-de-datos-SQL-para-MYSQL-SQL-Server-y-MS-Access/)

Primero veamos algo, en SQL se suele distinguir entre comandos y cláusulas, los comandos son las sentencias principales para realizar operaciones en la base de datos, mientras que las cláusulas son comandos que se utilizan en conjunto con las sentencias para proporcionar detalles adicionales sobre cómo se deben realizar estas operaciones.

Es importante recordar que, dentro de las categorías de comandos que utiliza un Data Scientist, se encuentran los de DQL y DML. A continuación, se presentan algunas de las principales sentencias de estos comandos:



- SELECT: se utiliza para seleccionar y recuperar datos de una tabla o vista.
- INSERT: se utiliza para insertar nuevos datos en una tabla.
- UPDATE: se utiliza para actualizar/modificar los datos existentes en una tabla.
- DELETE: se utiliza para eliminar datos de una tabla.

Como se mencionó adicionalmente a esto existen cláusulas con las que se combinan las sentencias, cuya importancia es crucial para realizar ciertos filtros y órdenes dentro de las consultas:



- FROM: se utiliza para especificar la tabla o tablas de las cuales se desean recuperar los datos.
- WHERE: se utiliza para filtrar los datos que se recuperan en función de una o varias condiciones.
- GROUP BY: se utiliza para agrupar los datos en función de una o varias columnas.
- HAVING: se utiliza para filtrar los resultados de la agrupación basándose en una condición.
- ORDER BY: se utiliza para ordenar los datos seleccionados en función de una o varias columnas.
- JOIN: se utiliza para combinar datos de dos o más tablas en función de una columna común.

- INTO: para usar con INSERT
- SET: para usar con UPDATE

Te preguntarás por qué está siempre resaltada la cláusula "JOIN", esto es porque es crucial que comprendas la importancia de la cláusula "JOIN" en SQL, ya que tiene múltiples formas de utilizarse y existen distintos tipos de JOIN que son esenciales para el trabajo de un Data Scientist. Aunque es un tema clave, no debes preocuparte, ya que primero te familiarizarás con los comandos comunes de SQL para que puedas sumergirte en este apartado de manera exhaustiva. Una vez que domines las prácticas previas, tendrás la frutilla del postre, que es la comprensión completa de esta cláusula esencial.

2.6. Ejemplos de sintaxis DQL para comprender el funcionamiento de cada comando y cláusula

Suposición: Partimos de una tabla llamada "Estudiantes" con las columnas "Nombre", "Apellido", "DNI", "Telefono", "Provincia", "Edad"

Para consultar por todos los campos de una tabla:

```
SELECT * FROM Estudiantes;
```

Para consultar solo las columnas Nombre y Apellido:

```
SELECT Nombre, Apellido FROM Estudiantes;
```

Para consultar todos los campos de estudiantes de la Provincia de Córdoba:

```
SELECT * FROM Estudiantes  
WHERE Provincia = "Córdoba";
```

Para consultar todos los campos de estudiantes menos los de Provincia de Córdoba:

```
SELECT * FROM Estudiantes  
WHERE Provincia != "Córdoba";
```

Con el operador != o <> podemos ver los "distintos de"

Para ver los estudiantes cuyo apellido comience con "A":

```
SELECT * FROM Estudiantes  
WHERE Nombre like "A%";
```

En SQL, la función "LIKE" se utiliza en las consultas de búsqueda de datos para buscar patrones específicos en una columna de texto. La función "LIKE" se utiliza junto con comodines para hacer coincidir patrones de caracteres en la columna de texto.

El operador comodín más común utilizado con la función "LIKE" es el símbolo "%" (porcentaje), que puede representar cualquier cadena de caracteres de cualquier longitud. Por ejemplo, la consulta "SELECT * FROM Estudiantes WHERE nombre LIKE 'A%' recuperará todos los registros de la tabla de estudiantes que tengan nombres que comiencen con la letra "A".

En cambio si pongo el % adelante y atrás de la A va traer todos los registros que contengan una A en nombre independientemente en donde esté ubicada esa A.

Así como el LIKE está el "NOT LIKE", para ver todos aquellos caracteres que no tengan esa condición.

Un recurso que te ayudará a practicar con este tipo de búsquedas, denominadas "expresiones regulares", es el siguiente: [SQL LIKE Operator \(w3schools.com\)](https://www.w3schools.com/sql/sql_like.asp)

→ Para ver los estudiantes cuya edad sea mayor a 25 años:

```
SELECT * FROM Estudiantes  
WHERE Edad > 25;
```

Observa que para los números no es necesario establecer la condición entre comillas, esto es solo necesario para las cadenas de texto.

Como el operador > mayor que, podemos usar el < y adicionarle el = si queremos que nos traiga por ejemplo los mayores o iguales que 25 años hubiéramos colocado >= .

Estudia más ejemplos en este recurso:

- [SQL Operators \(w3schools.com\)](https://www.w3schools.com/sql/sql_operators.asp)
- [Operadores lógicos \(Transact-SQL\) - SQL Server | Microsoft Learn](https://learn.microsoft.com/es-es/sql/queries/queries-logical-operators-transact-sql)

Combinar filtros, quiero ver los estudiantes cuya edad sea igual a 25 y sean de la Provincia de Buenos Aires:

```
SELECT * FROM Estudiantes  
WHERE Edad =25 AND Provincia="Buenos Aires";
```

Si quiero aplicar un conjunto de condiciones, estudiantes de la Provincia de Buenos Aires, Salta y Mendoza:

```
SELECT * FROM Estudiantes  
WHERE Provincia IN ("Buenos Aires", "Salta", "Mendoza");
```

Ordenar estudiantes de manera descendente por edad: por default SQL al arrojar las consultas las ordena de manera ascendente:

```
SELECT Edad FROM Estudiantes  
ORDER BY Edad DESC
```

Renombrar tablas - Sentencia "AS":

```
SELECT Edad FROM Estudiantes AS E
```

Renombrar campos calculados que van a ir como columnas:

```
SELECT Edad AS 'Años' FROM Estudiantes
```

Aclaración: el AS solo maquilla, no modifica la tabla original, maquilla en las consultas para facilidad del analista en ese momento!

Guardar resultados de consultas de SQL

File - Save Results As - Guardar

Como un Data Scientist trabaja con grandes volúmenes de datos, es recomendable utilizar el comando TOP antes de realizar consultas en la base de datos. Al agregar el comando TOP y especificar un número entre paréntesis, se puede limitar la cantidad de datos que se muestran en los resultados de la consulta. Esto permite obtener una vista general de la base de datos que se está investigando:

```
SELECT TOP (10) *  
FROM Estudiantes;
```

CONSULTA COMODÍN: DISTINCT → Traer sin duplicados de registros enteros

```
SELECT DISTINCT *  
FROM Estudiantes;
```

CONSULTA COMODÍN: DISTINCT → Traer valores únicos todos los valores distintos de una columna

```
SELECT DISTINCT Nombres  
FROM Estudiantes;
```

Nunca se ejecuta sobre el campo de "Primary Key", entendiendo Primary Key como el "DNI" de cada tabla, nunca encontrará registros repetidos, para ver si hay alguno repetido no lo tengo que colocar dentro del SELECT.

2.7. Ejemplos de sintaxis DML para comprender el funcionamiento de cada comando y cláusula

Las sentencias de DML puede que no se puedan utilizar dependiendo la compañía y los permisos asignados al Data Scientist, pero existen casos que sí pueden realizarse estas modificaciones en vistas y es necesario conocerlas.

Insertar datos en una tabla

Como Data Scientist, la sentencia INSERT en SQL puede ser útil para agregar nuevos registros a una tabla, actualizar los registros existentes y consolidar datos de varias fuentes en una sola tabla de la base de datos SQL.

```
INSERT INTO Estudiantes ("Nombre", "Apellido")
```

```
values ("Valeria", "Pereyra");
```

En este ejemplo se podría hacer esto de agregar datos sólo en algunos campos solo sí al crearse la base de datos permite que el resto de los campos adquieran el valor NULL, sino dará error.

Modificación de registros

Como Data Scientist, la sentencia UPDATE en SQL puede ser útil para actualizar los valores existentes en una tabla de la base de datos SQL. Esto puede ser necesario para corregir errores, normalizar datos, limpiar datos y actualizar los datos en tiempo real en una aplicación.

```
UPDATE Estudiantes SET Provincia="Córdoba"
```

```
WHERE Provincia like "%Cor%";
```

Eliminar registros

Como Data Scientist, la sentencia DELETE en SQL puede ser útil para eliminar registros específicos de una tabla de la base de datos SQL. Esto puede ser necesario para limpiar datos, reducir el tamaño de la tabla, entre otros.

```
DELETE FROM Estudiantes WHERE Provincia like "%Santiago%";
```

Para eliminar todos los registros de la tabla, simplemente se elimina el WHERE de la sentencia.

2.8. Funciones de agregación en SQL

Además de los comandos básicos de DQL y DML, un data scientist también debería conocer las funciones de agregación en SQL, ya que son fundamentales para realizar análisis estadísticos y resúmenes de datos. Algunas de las funciones de agregación más comunes en SQL incluyen SUM, COUNT, AVG, MIN y MAX.

¿Cómo se definen las funciones de agregación en SQL?

Las funciones de agregación se definen en la cláusula SELECT de una consulta SQL y se aplican a una o varias columnas de la tabla seleccionada. Por ejemplo, la siguiente consulta SQL calcularía la suma de la columna "Importe" en la tabla "Compras":

```
SELECT SUM(Importe) FROM Compras
```

Cláusulas mencionadas en el DQL como GROUP BY y HAVING se utilizan junto a estas funciones de agregación, veamos cada caso:

1. Utilización de GROUP BY: para agrupar un conjunto de registros en función de los valores de una o más columnas.

Selección de cantidad de cursos que toma cada estudiante

```
SELECT Estudiantes, COUNT(Cursos) AS Cantidad  
FROM Estudiantes  
GROUP BY Estudiantes;
```

Además de las funciones de agregación SUM y COUNT, hay muchas otras funciones que pueden utilizarse en combinación con la cláusula GROUP BY. Aquí te presento algunas de las funciones de agregación más comunes en SQL:

- AVG: Calcula el promedio de los valores de una columna. Por ejemplo:

```
SELECT nombre, AVG(edad)  
FROM personas  
GROUP BY nombre;
```

- MIN/MAX: Retorna el valor mínimo/máximo de una columna. Por ejemplo:

```
SELECT categoria, MAX(precio)  
FROM productos  
GROUP BY categoria;
```

- SUM: Retorna la suma de los valores de una columna.

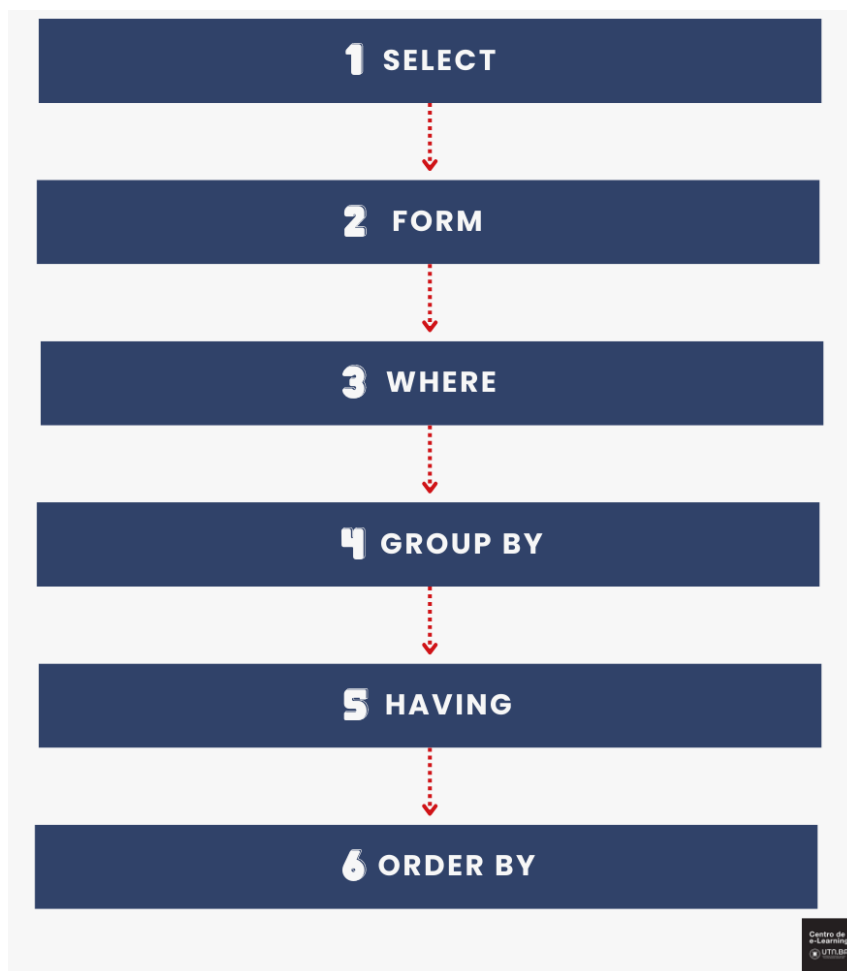
2. Utilización de HAVING: Solamente si estamos hablando de funciones de agregación, permite operar sobre campos que fueron generados a partir de una función.

Selección de cantidad de cursos que toma cada estudiante aquellos que toman más de 3 cursos:

```
SELECT Estudiantes, COUNT(Cursos) AS Cantidad  
FROM Estudiantes  
GROUP BY Estudiantes  
HAVING COUNT (Cursos) >3;
```

** Importante siempre el HAVING va después del GROUP BY, y antes de la cláusula ORDER BY (si es que se utiliza).

Recordar, el orden de escritura, ¡sino el software dará un error!



2.9. Funciones escalares

Las funciones escalares en SQL son un catálogo de operaciones que nos facilitan la realización de cálculos y transformaciones de los datos.

Investigá todas aquí: [SQL Server Functions \(w3schools.com\)](https://www.w3schools.com/sql/)

Algunas de las más importantes para un Data Scientist son:

1. CAST(): La función CAST() se utiliza para convertir un tipo de datos en otro. Por ejemplo, se puede usar para convertir una cadena en un número, lo que es útil en el análisis de datos.
2. DATE(): La función DATE() se utiliza para extraer la parte de fecha de una columna de fecha y hora. Esto es útil en el análisis de datos temporales.
3. CONCAT(): Unir campos o caracteres
4. REPLACE(): Reemplazar caracteres en un campo
5. TRIM(): Eliminación de espacios en blanco al inicio y final del valor de una celda
6. GETDATE() es una función de fecha y hora en SQL que devuelve la fecha y hora actual del sistema.
7. DATEDIFF() función para calcular tiempos transcurridos entre dos fechas.

Te recomiendo que no te memorices ninguna función ni comando, sino que siempre tengas a mano la documentación y puedas verificar cada sentencia antes de aplicarla junto con sus requerimientos!

Ejemplos en clases:

1. tabla "ventas" con la siguiente estructura:

```
| id_venta | fecha | total |
```

```
SELECT fecha, CAST(total AS FLOAT) as total_num FROM ventas;
```

2. En esta tabla, la columna "fecha" es de tipo datetime (fecha y hora), pero necesitamos agrupar las ventas por día para nuestro análisis. Para ello, es necesario extraer la fecha (sin la hora) de la columna "fecha", lo cual podemos hacer utilizando la función DATE() de la siguiente manera:

```
SELECT DATE(fecha) as fecha_sin_hora, SUM(total) as total_diario
```

```
FROM ventas
```

```
GROUP BY DATE(fecha);
```

3. | id_empleado | nombre | apellido_paterno | apellido_materno | fecha_nacimiento |

En esta tabla, queremos concatenar los nombres y apellidos de los empleados en una sola columna, pero también queremos reemplazar los acentos y caracteres especiales por su equivalente sin acentos. Para ello, podemos utilizar las funciones CONCAT() y REPLACE() de la siguiente manera:

```
SELECT CONCAT(REPLACE(nombre, 'á', 'a'),
```

```
' ', REPLACE(apellido_paterno, 'é', 'e'),' ', REPLACE(apellido_materno, 'í', 'i')) as nombre_completo
```

```
FROM empleados;
```

4. DateDiff basandome en la fecha de nacimiento de los empleados

```
SELECT nombre, apellido_paterno, apellido_materno, fecha_nacimiento,
```

```
DATEDIFF(CURRENT_DATE(), fecha_nacimiento) / 365 AS edad
```

```
FROM empleados;
```

2.10. Subconsultas

Las subconsultas son un tipo de función anidada, que se utiliza para filtrar los resultados de la consulta principal, utilizando información de otra tabla o de la misma tabla. La subconsulta puede ser escrita en cualquier lugar donde se acepte una expresión SQL, como en la cláusula WHERE, la cláusula HAVING o la cláusula SELECT.

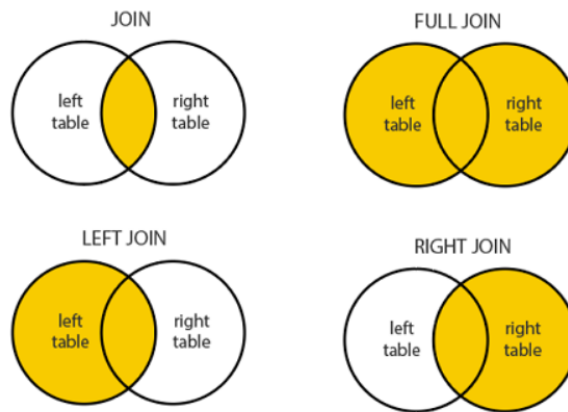
Para entenderlo mejor, veamos un ejemplo con la base de datos que utilizamos para ejercitar la primer parte de consultas en esta unidad:

[SUBCONSULTAS](#)

2.11. ¿Qué es un JOIN?

La cláusula JOIN es utilizada en SQL para combinar registros de diferentes tablas basándose en una condición de unión. Esta condición de unión generalmente involucra la clave primaria de una tabla y la clave externa (o foránea) de otra tabla. La cláusula JOIN se utiliza para obtener información que se encuentra en diferentes tablas, y permite relacionar los datos de estas tablas para obtener resultados más significativos.

Hay diferentes tipos de JOIN disponibles en SQL, incluyendo INNER JOIN, LEFT JOIN, RIGHT JOIN y FULL OUTER JOIN. Cada uno de ellos tiene una sintaxis específica y se utiliza para diferentes propósitos. Además, la cláusula JOIN se combina con la cláusula ON para especificar la condición de unión que se utilizará para combinar los datos de las tablas.



1. **INNER JOIN:** Este tipo de JOIN devuelve únicamente los registros que tienen correspondencia en ambas tablas. En otras palabras, los registros que cumplen con la condición de unión especificada en la cláusula ON. Si un registro no tiene correspondencia en la otra tabla, no se incluirá en los resultados.
2. **LEFT JOIN:** Este tipo de JOIN devuelve todos los registros de la tabla de la izquierda (la tabla que se menciona antes de la cláusula JOIN), junto con los registros correspondientes de la tabla de la derecha (la tabla que se menciona después de la cláusula JOIN). Si un registro no tiene correspondencia en la tabla de la derecha, se devolverá NULL en lugar de los valores correspondientes.
3. **RIGHT JOIN:** Este tipo de JOIN es similar al LEFT JOIN, pero devuelve todos los registros de la tabla de la derecha y los registros correspondientes de la tabla de la izquierda. Si un registro no tiene correspondencia en la tabla de la izquierda, se devolverá NULL en lugar de los valores correspondientes.
4. **FULL OUTER JOIN:** Este tipo de JOIN devuelve todos los registros de ambas tablas, incluyendo los registros que no tienen correspondencia en la otra tabla. Si un registro no tiene correspondencia en la otra tabla, se devolverá NULL en lugar de los valores correspondientes.

Veamos juntos el siguiente recurso para entender desde un abordaje visual y conociendo la sintaxis: [SQL Joins \(w3schools.com\)](https://www.w3schools.com/sql/joins.asp)

2.12. Automatizaciones dentro de SQL

Como Data Scientist podemos querer generar automatizaciones sobre ciertas consultas, para esto se utilizan procedimientos, en la jerga de SQL denominados "Stored procedure", que son procedimientos almacenados o programas que permiten realizar una o varias tareas dentro de una o varias bases al mismo tiempo. Permitiendo automatizar así las tareas y reutilizar código, sin tener la necesidad de generar las sentencias cada vez.

Creación de procedimiento:

```
CREATE PROCEDURE
```

```
SelecionarEstudiantes
```

```
AS
```

```
SELECT * FROM Estudiantes;
```

```
Ejecutar procedimiento
```

```
EXEC
```

```
SelecionarEstudiantes;
```

¡Estudiantes, están listos para dar un gran paso en su camino como futuros Data Scientists! En este tema, hemos aprendido sobre las bases de datos más pequeñas y los conceptos básicos de SQL, ¡pero no subestimen su importancia! Esta introducción es enriquecedora y esencial para que puedan comprender y manejar los comandos necesarios para acceder y manipular grandes conjuntos de datos en su futuro rol como Data Scientists.

¡Imaginen todas las posibilidades que tendrán una vez que dominen SQL! Este lenguaje es uno de los más demandados en la industria para este trabajo, su fácil sintaxis y gran comunidad lo convierten en una herramienta valiosa para cualquier profesional de datos.

3. Exploración y Visualización de datos y el rol de las herramientas de visualización más valoradas en la industria: Tableau y Power BI

En la actualidad, los científicos de datos están encargados de analizar grandes cantidades de datos y transformarlos en información valiosa para la toma de decisiones empresariales. En este proceso, las herramientas de visualización de datos son fundamentales para explorar y presentar los resultados de manera clara y efectiva. Entre las herramientas más valoradas en la industria se encuentran Tableau y Power BI, ambas con una gran cantidad de características y funcionalidades que permiten a los científicos de datos obtener insights precisos y relevantes.

Tanto Power BI como Tableau son herramientas de inteligencia empresarial o BI por sus siglas en inglés, que se basan en la visualización de datos, permitiendo a los usuarios crear gráficos y tableros interactivos para comunicar información de manera efectiva. Estas herramientas son útiles para los data scientists porque les permiten explorar y visualizar datos de manera eficiente, lo que puede ayudar a identificar patrones, tendencias y relaciones entre los datos.

En este capítulo, exploramos una visión general de estas herramientas, revisaremos sus características principales, fortalezas y debilidades, y cómo seleccionar la herramienta adecuada según las necesidades de la empresa y los objetivos de análisis de datos.

¿Tableau o Power BI?

Primero veamos como se definen ambos softwares:

Desde su página oficial, Tableau se define como una plataforma de análisis y visualización de datos que permite a las personas y organizaciones ver y comprender sus datos. Según la página web de Tableau, "Tableau ayuda a cualquier persona a analizar, visualizar y compartir información de manera más eficiente y efectiva".

Mientras que Power BI se define como una plataforma de análisis de negocios de Microsoft que permite a las organizaciones conectarse a múltiples fuentes de datos, transformar y modelar los datos, crear visualizaciones interactivas y compartir sus análisis con otros miembros de la organización.

¿Entonces podemos decir que usar Tableau y Power BI es lo mismo?

Aunque ambas herramientas comparten algunas características y se utilizan para lograr objetivos similares, no podemos decir que usar Tableau y Power BI es lo mismo. Veamos algunas diferencias:

Tableau se destaca por su enfoque en el "Data Storytelling", lo que significa que no solo se enfoca en la visualización de datos, sino que también ayuda a las personas a comunicar historias efectivas a través de sus datos. Mientras que, Power BI se enfoca en la colaboración y el trabajo en equipo, permitiendo a los usuarios compartir informes y paneles de control con otros miembros de la organización.

Veamos un poco más sobre sus diferencias y similitudes

Característica	Tableau	Power BI
Creación	Por Tableau Software USA (2003)	Microsoft (2013)
Precio	Gratuito para estudiantes, \$12-70 por usuario al mes para planes empresariales	Gratuito para estudiantes, \$9.99-20.00 por usuario al mes para planes empresariales
Integración de datos	Permite la integración de datos de múltiples fuentes con facilidad	Ofrece una variedad de conectores de datos, pero la integración de datos de múltiples fuentes puede requerir cierto conocimiento técnico
Visualizaciones	Amplia gama de tipos de visualización y opciones de personalización	Menos tipos de visualización que Tableau, pero más fáciles de usar para los principiantes
Funcionalidad de modelado de datos	No tiene una funcionalidad de modelado de datos incorporada, pero se puede integrar con otras herramientas para esta tarea	Ofrece una funcionalidad de modelado de datos incorporada y una experiencia de usuario más orientada a la creación de informes
Escalabilidad	Puede manejar grandes conjuntos de datos, pero requiere una infraestructura robusta	Tiene una arquitectura escalable y puede manejar grandes cantidades de datos de manera eficiente
Aprendizaje y soporte	La curva de aprendizaje es un poco más empinada que Power BI, pero ofrece una amplia gama de recursos y una comunidad activa de usuarios	Ofrece una experiencia de usuario más intuitiva para los principiantes y una gran cantidad de recursos y soporte en línea
Plataformas compatibles	Disponible en Windows y Mac y en la nube	Disponible en Windows y también en la nube

Tanto Tableau como Power BI son herramientas de visualización de datos muy potentes y populares en el mundo de la ciencia de datos. Ambas herramientas tienen características y funcionalidades que los profesionales de la ciencia de datos pueden encontrar útiles y valiosas en su trabajo.

Sin embargo, la elección entre Tableau y Power BI depende en gran medida de las necesidades y preferencias específicas de cada proyecto y organización. Algunos factores a considerar al decidir entre estas herramientas pueden incluir:

1. Volumen de datos: Si se trabaja con grandes volúmenes de datos, Power BI puede ser una mejor opción, ya que tiene una arquitectura escalable y puede manejar grandes cantidades de datos de manera eficiente.
2. Integración de datos: Si se necesita integrar datos de diferentes fuentes, Tableau puede ser una mejor opción, ya que tiene una funcionalidad de integración de datos más robusta.
3. Experiencia del usuario: Ambas herramientas tienen interfaces fáciles de usar, pero algunas personas pueden encontrar una herramienta más fácil de usar que la otra. Por lo tanto, se puede optar por la que mejor se adapte a sus necesidades y habilidades.
4. Costo: Tableau y Power BI tienen diferentes planes de precios, por lo que es importante evaluar qué plan se adapta mejor a las necesidades y presupuesto del proyecto.

¡Es hora de poner manos a la obra! En este proceso, exploramos la interfaz de cada herramienta, sus principales visualizaciones y te proporcionaremos un roadmap personalizado para aprender más sobre cada software, en función de tus intereses como futuro Data Scientist. De esta manera, podrás adquirir una comprensión profunda de cada herramienta y estar mejor preparado para enfrentar los desafíos en el campo de la ciencia de datos.

3.1. Acciones prácticas en Power BI y Tableau que se aprovechan desde el rol del data scientist.

Acciones prácticas en Power BI y gráficas que se aprovechan desde el rol del data scientist.

1. Importar datos desde diferentes fuentes, como Excel o CSV, utilizando Power Query.
2. Crear informes y paneles utilizando diferentes tipos de visualizaciones, como gráficos de barras, gráficos circulares, gráficos de líneas y gráficos de dispersión.
3. Analizar los datos utilizando herramientas de análisis de Power BI, como la segmentación de datos y la creación de medidas personalizadas utilizando DAX.
4. Utilizar herramientas avanzadas de visualización como mapas y gráficos de Gantt para presentar datos de manera más efectiva.
5. Aprender a automatizar tareas comunes utilizando Power Automate.

Acciones prácticas en Tableau y gráficas que se aprovechan desde el rol del data scientist.

1. Importar datos desde diferentes fuentes, como Excel o CSV, utilizando la función "Conectar a datos" en Tableau.
2. Crear un panel de control utilizando diferentes tipos de visualizaciones, como gráficos de barras, gráficos circulares, gráficos de líneas y gráficos de dispersión.
3. Utilizar herramientas de análisis de Tableau, como filtros y cálculos, para explorar los datos y obtener información valiosa.
4. Utilizar herramientas avanzadas de visualización como mapas y gráficos de embudo para presentar datos de manera más efectiva.
5. Aprender a compartir y publicar paneles y visualizaciones utilizando Tableau Server o Tableau Online.

Gráficos útiles para el rol de un Data Scientist

1. Histogramas y diagramas de caja y bigotes para entender la distribución de los datos y detectar valores atípicos.
2. Gráficos de correlación, como los diagramas de dispersión, para visualizar las relaciones entre dos o más variables.
3. Gráficos de series de tiempo, como los gráficos de líneas, para visualizar la evolución de una variable a lo largo del tiempo.
4. Mapas de calor y mapas de choropleth para visualizar datos geoespaciales y detectar patrones regionales.
5. Diagramas de árbol y diagramas de flujo para visualizar relaciones complejas entre múltiples variables.

Ejemplo de dashboards en la industria e interpretación en clases

- En Power BI: [Power BI Playground - Showcases](#)
- En Tableau: [Discover | Tableau Public](#)

4. Conclusiones

En este capítulo, hemos explorado el fascinante mundo de la extracción de datos y el poderoso rol que juega SQL. También hemos aprendido cómo las herramientas de visualización como Tableau y Power BI pueden transformar datos crudos en información valiosa para la toma de decisiones estratégicas.

Pero esto es solo el comienzo, porque en el próximo capítulo, ¡vamos a llevar tus habilidades de Data Science al siguiente nivel! Aprenderás a aplicar herramientas informáticas en la fase de análisis de datos utilizando Python, una de las herramientas más poderosas y versátiles en el mundo de la Ciencia de Datos.

No puedes perder la oportunidad de dominar estas habilidades esenciales para destacarte en la industria de la Ciencia de Datos. ¡Así que prepárate para un emocionante viaje mientras seguimos explorando el ciclo de vida de los datos con herramientas informáticas!

5. Bibliografía utilizada y sugerida

En esta unidad se ha dado un vistazo general de algunas herramientas informáticas ampliamente utilizadas en el ciclo de vida de la ciencia de datos. Para profundizar en el uso de SQL en la gestión de bases de datos, se puede consultar el tutorial en línea de W3Schools o el libro "Beginning SQL" de Wilton y Colby. Además, en el documento "Modelado de una Base de Datos (DER - MR)" de Palomares se encuentra información importante sobre el modelado de bases de datos. Para la visualización de datos, se han revisado dos herramientas importantes: Power BI y Tableau. En el libro "Introducing Power BI" de Clark, se ofrece una introducción práctica a Power BI y sus funcionalidades, mientras que en el capítulo "Data Visualization using Tableau" del libro "Fundamentals of Data Science" de Wagh, Bhende y Thakare, se presenta una guía para la visualización efectiva de datos utilizando Tableau. Por último, las páginas web de Microsoft Power BI y Tableau contienen información valiosa sobre estas herramientas y sus funcionalidades.

1. Clark, D. (2020). Introducing power BI. In Beginning Microsoft Power BI (pp. 1–20). Apress. http://dx.doi.org/10.1007/978-1-4842-5620-6_1
2. Modelado de una Base de Datos (DER - MR) - Modelado de una Base de Datos Autor: Alfonso Palomares. (n.d.). Studocu. Retrieved March 11, 2023, from <https://www.studocu.com/es-mx/document/universidad-virtual-del-estado-de-guanajuato/base-de-datos/modelado-de-una-base-de-datos-der-mr/26181834>
3. SQL tutorial. (n.d.). Retrieved March 16, 2023, from <https://www.w3schools.com/sql/>
4. Tableau: Business intelligence and analytics software. (n.d.). Tableau. Retrieved March 17, 2023, from <https://www.tableau.com/>
5. Visualización de datos. (n.d.). Microsoft Power BI. Retrieved March 17, 2023, from <https://powerbi.microsoft.com/es-es/>
6. Wagh, S. J., Bhende, M. S., & Thakare, A. D. (2021). Data visualization using tableau. In Fundamentals of Data Science (pp. 249–267). Chapman and Hall/CRC. <http://dx.doi.org/10.1201/9780429443237-14>
7. Wilton, P., & Colby, J. (2005). Beginning SQL. John Wiley & Sons.