# Curso de SQL Server

Aprende a administrar bases de datos con SQL Server

Empezar

## Descripción general

Este curso te proporcionará los conocimientos necesarios para administrar bases de datos utilizando SQL Server. Aprenderás desde los conceptos básicos de bases de datos hasta consultas SQL SELECT, funciones de agregación, modificación de datos

y manipulación de relaciones entre tablas. También explorarás el uso de subconsultas y consultas anidadas para obtener información más compleja de tus bases de datos.

01 Introducción

# Introducción a SQL Server

01 | Introducción a SQL Server

SQL Server es un sistema de administración de bases de datos relacional desarrollado por Microsoft. Es una herramienta poderosa que permite almacenar, gestionar y manipular grandes cantidades de datos en forma estructurada.

## Ediciones de SQL Server

SQL Server viene en varias ediciones, desde versiones gratuitas hasta empresariales. Esto significa que se adapta a diferentes necesidades y presupuestos. Algunas de las ediciones más comunes son:

- **SQL Server Express**: Es una versión gratuita de SQL Server que tiene limitaciones en cuanto a escala, rendimiento y funcionalidades. Sin embargo, es una excelente opción para proyectos pequeños o para aprender sobre SQL Server.
- **SQL Server Standard**: Esta edición ofrece una amplia gama de funcionalidades y capacidades para el desarrollo y administración de bases de datos. Es adecuada para la mayoría de las aplicaciones empresariales.
- **SQL Server Enterprise**: Es la edición más completa, diseñada para aplicaciones y sistemas de misión crítica que requieren un alto rendimiento, escalabilidad y disponibilidad. Esta edición incluye todas las funcionalidades de SQL Server.

## Herramientas de administración y desarrollo

SQL Server cuenta con una variedad de herramientas que facilitan la administración y el desarrollo de bases de datos. Una de las más utilizadas es el **SQL Server Management Studio (SSMS)**. Esta herramienta proporciona una interfaz gráfica intuitiva para administrar y manipular bases de datos y ofrece capacidades avanzadas como la creación de consultas, la visualización de planes de ejecución y la depuración de código.

Además de SSMS, también existen otras herramientas y aplicaciones complementarias que amplían las capacidades de SQL Server, como el **SQL Server Data Tools (SSDT)** para la creación de proyectos de bases de datos, y el **SQL Server Profiler** para el análisis y monitoreo del rendimiento de las consultas.

En resumen, SQL Server es una potente herramienta de administración de bases de datos relacional desarrollada por Microsoft. Viene en varias ediciones para adaptarse a diferentes necesidades y presupuestos, y cuenta con herramientas de administración y desarrollo que facilitan la gestión y manipulación de bases de datos.

Conocer y dominar SQL Server es fundamental para cualquier persona interesada en el manejo eficiente y efectivo de datos en entornos empresariales.

Conclusión - Introducción a SQL Server

En este curso de SQL Server, hemos aprendido los conceptos básicos de esta plataforma de administración de bases de datos. Desde la introducción a SQL Server, pasando por los fundamentos de las bases de datos y las consultas SQL SELECT, hasta llegar a las funciones de agregación y la modificación de datos con INSERT, UPDATE y DELETE. También hemos explorado los JOINs y las relaciones entre tablas, así como las subconsultas y consultas anidadas. Con este conocimiento, los estudiantes estarán preparados para trabajar con eficiencia y solvencia en el manejo de bases de datos en SQL Server.

# Conceptos Básicos de Bases de Datos



Una base de datos es un contenedor organizado para almacenar datos en tablas. Las tablas están formadas por filas y columnas. En este módulo, exploraremos los conceptos básicos de las bases de datos y cómo trabajar con ellas en SQL Server.

## Estructura de una Base de Datos

Una base de datos está compuesta por tablas, que son estructuras de datos que almacenan los registros. Cada tabla se organiza en filas y columnas.

- **Filas**: Cada fila representa un registro en la tabla. Por ejemplo, en una tabla de empleados, cada fila podría representar a un empleado con su información personal.
- **Columnas**: Cada columna representa un atributo o campo de datos. Por ejemplo, en una tabla de empleados, las columnas pueden ser el nombre, la edad y el salario.

# Tipos de Datos

Los tipos de datos en una base de datos definen el tipo de información que se puede almacenar en una columna. Algunos ejemplos comunes de tipos de datos son:

- **Texto**: Almacena cadenas de caracteres como nombres, direcciones o descripciones.
- **Numérico**: Almacena valores numéricos como números enteros o decimales.
- **Fecha y Hora**: Almacena fechas y/o horas.
- **Booleano**: Almacena valores verdadero o falso, representando estados lógicos.

Es importante utilizar el tipo de dato adecuado para cada columna, ya que esto garantiza la integridad y consistencia de los datos almacenados.

# Claves Primarias y Claves Foráneas

Las claves primarias y las claves foráneas son conceptos fundamentales en las bases de datos relacionales, como SQL Server. Estas claves establecen relaciones entre las tablas y ayudan a garantizar la integridad de los datos.

- **Clave Primaria**: Es un atributo único en una tabla que identifica de manera exclusiva cada registro. Por ejemplo, en una tabla de alumnos, el número de identificación del alumno podría ser la clave primaria.
- **Clave Foránea**: Es una columna en una tabla que hace referencia a la clave primaria de otra tabla. Establece una relación entre las tablas. Por ejemplo, en una tabla de calificaciones, la clave foránea sería el número de identificación del alumno.

Establecer correctamente las claves primarias y foráneas es esencial para garantizar la integridad de los datos y permitir consultas y operaciones más eficientes.

¡Continúa aprendiendo sobre SQL Server y estarás en camino de convertirte en un experto en bases de datos!

> ***Recuerda: La práctica constante y la experimentación son clave para dominar SQL Server y las bases de datos relacionales en general.***

Continúa explorando los conceptos básicos de SQL Server y descubre cómo realizar consultas SELECT en el próximo módulo. ¡Sigue adelante!

Conclusión - Conceptos Básicos de Bases de Datos

En este módulo de Conceptos Básicos de Bases de Datos, hemos explorado los fundamentos de las bases de datos. Hemos aprendido cómo organizar y almacenar datos en tablas, utilizando filas y columnas. También hemos comprendido la importancia de los tipos de datos para definir la información que se puede almacenar y de las claves primarias y foráneas para establecer relaciones entre tablas. Estos conceptos son esenciales para el correcto diseño y gestión de bases de datos en SQL Server.

# Consultas SQL SELECT

03 | Consultas SQL SELECT

La sentencia **SELECT** se utiliza en SQL para recuperar datos de una tabla específica. Esta sentencia es esencial en la consulta y manipulación de información en una base de datos relacional.

## Sintaxis básica

La sintaxis básica de una consulta **SELECT** es la siguiente:

```
 SELECT columnas
FROM tabla
```

Donde:

- **columnas** representa el conjunto de columnas que deseamos obtener en los resultados de la consulta. Podemos seleccionar una o varias columnas separadas por comas.

- **tabla** es el nombre de la tabla de la cual queremos obtener los datos.

## Filtrado de resultados

Podemos filtrar los resultados de una consulta utilizando la cláusula **WHERE**. Esta cláusula nos permite especificar una condición que deben cumplir los registros para ser considerados en los resultados.

La sintaxis básica de la cláusula **WHERE** es la siguiente:

```
 SELECT columnas
FROM tabla
WHERE condición
```

Donde:

- **condición** es una expresión lógica que determina qué registros cumplen con los criterios establecidos.

## Ordenamiento de resultados

Podemos ordenar los resultados de una consulta utilizando la cláusula **ORDER BY**. Esta cláusula nos permite especificar la columna por la cual queremos ordenar los registros, ya sea en orden ascendente (ASC) o descendente (DESC).

La sintaxis básica de la cláusula **ORDER BY** es la siguiente:

```
 SELECT columnas
FROM tabla
ORDER BY columna [ASC|DESC]
```

Donde:

- **columna** es el nombre de la columna por la cual queremos ordenar los registros.
- **ASC** indica que los registros deben ser ordenados en orden ascendente (de menor a mayor).
- **DESC** indica que los registros deben ser ordenados en orden descendente (de mayor a menor).

## Ejemplos de consultas SELECT

A continuación se presentan algunos ejemplos de consultas **SELECT**:

1. Obtener todos los registros de una tabla:

```
 SELECT *
FROM tabla
```

2. Obtener solo los nombres y edades de los empleados mayores de 30 años:

```
 SELECT nombre, edad
FROM empleados
WHERE edad > 30
```

3. Obtener los registros de una tabla ordenados por el campo fecha de forma descendente:

```
 SELECT *
FROM tabla
ORDER BY fecha DESC
```

4. Obtener los nombres y salarios de los empleados de un departamento específico:

---

```
 SELECT nombre, salario
FROM empleados
WHERE departamento = 'Ventas'
```

La sentencia **SELECT** es una herramienta poderosa para obtener y filtrar datos en una base de datos relacional. Con su utilización adecuada, podemos obtener la información necesaria para realizar consultas más complejas y avanzadas.

Conclusión - Consultas SQL SELECT

En esta sección sobre Consultas SQL SELECT, hemos adquirido los conocimientos necesarios para recuperar datos de una tabla utilizando la sentencia SELECT. Hemos aprendido cómo aplicar filtros utilizando la cláusula WHERE y cómo ordenar los resultados con ORDER BY. Estos son fundamentos básicos que nos permitirán realizar consultas eficientes y obtener la información deseada de manera precisa en SQL Server.

# Funciones de Agregación

04 | Funciones de Agregación

Las funciones de agregación son herramientas fundamentales en SQL Server que nos permiten realizar cálculos en conjuntos de datos. Estas funciones pueden aplicarse a columnas numéricas y devolver resultados como el total de registros, la suma de valores, el promedio, el mínimo y el máximo.

A continuación, presentaremos las principales funciones de agregación disponibles en SQL Server:

## COUNT

La función COUNT se utiliza para contar el número de registros en una tabla o columna en particular. Por ejemplo, si queremos saber cuántos clientes tenemos en nuestra base de datos, podemos utilizar esta función de la siguiente manera:

```
 SELECT COUNT(*) AS TotalClientes
FROM clientes;
```

En este ejemplo, estamos utilizando el asterisco (*) como parámetro de COUNT, lo que indica que queremos contar todos los registros de la tabla "clientes". El resultado se mostrará en una columna llamada "TotalClientes".

## SUM

La función SUM nos permite obtener la suma de los valores de una columna numérica. Supongamos que tenemos una tabla "ventas" con una columna "monto" que almacena el valor de cada venta realizada. Si queremos conocer el total de ventas realizadas, podemos utilizar la función SUM de esta manera:

```
 SELECT SUM(monto) AS TotalVentas
FROM ventas;
```

En este caso, estamos sumando el valor de la columna "monto" de la tabla "ventas" y mostrando el resultado en una columna llamada "TotalVentas".

## AVG

La función AVG se utiliza para calcular el promedio de los valores de una columna numérica. Si tenemos una tabla "productos" con una columna "precio" que almacena el precio de cada producto, podemos obtener el precio promedio utilizando la función AVG de la siguiente manera:

```
 SELECT AVG(precio) AS PrecioPromedio
FROM productos;
```

En este ejemplo, estamos calculando el promedio de la columna "precio" de la tabla "productos" y mostrando el resultado en una columna llamada "PrecioPromedio".

# MIN

La función MIN nos permite obtener el valor mínimo de una columna. Por ejemplo, si tenemos una tabla "empleados" con una columna "salario" que almacena los salarios de los empleados, podemos obtener el salario mínimo utilizando la función MIN de esta manera:

```
 SELECT MIN(salario) AS SalarioMinimo
FROM empleados;
```

En este caso, estamos seleccionando el valor mínimo de la columna "salario" de la tabla "empleados" y mostrándolo en una columna llamada "SalarioMinimo".

# MAX

La función MAX nos permite obtener el valor máximo de una columna. Siguiendo con el ejemplo anterior, si queremos conocer el salario máximo de los empleados, podemos utilizar la función MAX de la siguiente forma:

```
 SELECT MAX(salario) AS SalarioMaximo
FROM empleados;
```

Aquí estamos seleccionando el valor máximo de la columna "salario" de la tabla "empleados" y mostrándolo en una columna llamada "SalarioMaximo".

Estas funciones de agregación son una herramienta poderosa para realizar cálculos sobre conjuntos de datos en SQL Server. Su correcta utilización nos permitirá obtener información valiosa y realizar análisis más precisos en nuestras bases de datos.

Conclusión - Funciones de Agregación

En este apartado de Funciones de Agregación, hemos explorado las funciones COUNT, SUM, AVG, MIN y MAX, las cuales nos permiten realizar cálculos en conjuntos de datos. Además, hemos comprendido cómo utilizar la cláusula GROUP BY para agrupar resultados y la cláusula HAVING para filtrar grupos. Estas herramientas son fundamentales para el análisis y la obtención de información resumida en SQL Server.

# Modificación de Datos (INSERT, UPDATE, DELETE)

## INSERT

La sentencia INSERT se utiliza en SQL Server para insertar nuevos registros en una tabla. Permite especificar los valores que se desean insertar para cada columna de la tabla. A continuación, se muestra la sintaxis básica de la sentencia INSERT:

```
 INSERT INTO nombre_tabla (columna1, columna2, columna3, ...)
VALUES (valor1, valor2, valor3, ...);
```

Por ejemplo, supongamos que tenemos una tabla llamada "clientes" con las columnas "id_cliente", "nombre" y "correo_electronico". Podríamos insertar un nuevo cliente con el siguiente comando:

```
 INSERT INTO clientes (id_cliente, nombre, correo_electronico)
VALUES (1, 'Juan Pérez', 'juan@example.com');
```

## UPDATE

La sentencia UPDATE se utiliza para actualizar los datos existentes en una tabla. Permite modificar los valores de una o varias columnas en función de una condición específica. A continuación, se muestra la sintaxis básica de la sentencia UPDATE:

```
 UPDATE nombre_tabla
SET columna1 = valor1, columna2 = valor2, columna3 = valor3, ...
WHERE condicion;
```

Por ejemplo, supongamos que queremos actualizar el correo electrónico del cliente con id_cliente = 1. Podríamos hacerlo de la siguiente manera:

```
 UPDATE clientes
SET correo_electronico = 'nuevo_correo@example.com'
WHERE id_cliente = 1;
```

## DELETE

La sentencia DELETE se utiliza para eliminar uno o varios registros de una tabla. Permite eliminar tanto registros específicos como todos los registros de una tabla. A continuación, se muestra la sintaxis básica de la sentencia DELETE:

```
 DELETE FROM nombre_tabla
WHERE condicion;
```

Por ejemplo, supongamos que queremos eliminar todos los clientes que no tienen un correo electrónico registrado. Podríamos hacerlo de la siguiente manera:

```
 DELETE FROM clientes
WHERE correo_electronico IS NULL;
```

Recuerda tener cuidado al utilizar la sentencia DELETE, ya que una vez eliminados los registros, no se pueden recuperar automáticamente.

En resumen, la modificación de datos en SQL Server se realiza a través de las sentencias INSERT, UPDATE y DELETE. Con estas sentencias, puedes insertar nuevos registros en una tabla, actualizar los valores existentes y eliminar registros según ciertas condiciones. Estas operaciones son fundamentales para el mantenimiento de una base de datos y la actualización de la información contenida en ella.

Conclusión - Modificación de Datos (INSERT, UPDATE, DELETE)

En esta sección de Modificación de Datos, hemos aprendido a insertar nuevos registros con la sentencia INSERT, actualizar datos existentes con la sentencia UPDATE y eliminar registros con la sentencia DELETE. Estas operaciones son fundamentales para mantener la integridad y actualización de los datos en una base de datos en SQL Server.

# Joins y Relaciones

06 | Joins y Relaciones

En SQL Server, los "joins" y las relaciones son fundamentales para combinar datos de múltiples tablas y establecer conexiones entre ellas. Los "joins" nos permiten

combinar registros de diferentes tablas basándonos en una columna común, mientras que las relaciones definen cómo se relacionan las tablas entre sí.

## Tipos de Joins

Existen diferentes tipos de "joins" que se pueden utilizar en SQL Server:

1. **INNER JOIN**: Este tipo de "join" devuelve los registros que tienen coincidencias en ambas tablas. Es decir, solo mostrará los registros donde el valor de la columna que se está utilizando como criterio de combinación existe en ambas tablas.
2. **LEFT JOIN**: Un "left join" devuelve todos los registros de la tabla izquierda (la primera tabla escribida en la consulta) y los registros coincidentes de la tabla derecha (la segunda tabla escrita en la consulta). En caso de que no haya coincidencias en la segunda tabla, se mostrará NULL para las columnas de la tabla derecha.
3. **RIGHT JOIN**: A diferencia del "left join", un "right join" devuelve todos los registros de la tabla derecha y los registros coincidentes de la tabla izquierda. Si no hay coincidencias en la tabla izquierda, se mostrará NULL para las columnas de la tabla izquierda.

## Relaciones entre Tablas

Además de los "joins", las relaciones entre tablas son esenciales para establecer la conexión lógica y estructural entre ellas. Estas relaciones se definen a través de claves primarias y foráneas.

1. **Clave Primaria (Primary Key)**: Una clave primaria es una columna (o un conjunto de columnas) que identifica de manera única cada registro en una tabla. Se utiliza para evitar duplicados y garantizar la integridad de los datos. Cada tabla debe tener una única clave primaria.
2. **Clave Foránea (Foreign Key)**: Una clave foránea es una columna (o un conjunto de columnas) en una tabla que establece una relación con la clave primaria de otra tabla. La

clave foránea crea un vínculo entre las tablas, permitiendo el acceso a datos relacionados entre ellas.

## La Importancia de los Joins y las Relaciones

Los "joins" y las relaciones son cruciales en SQL Server ya que nos permiten obtener resultados más completos y precisos al combinar datos de diferentes tablas. Esto es especialmente útil cuando necesitamos extraer información de múltiples fuentes y realizar consultas más complejas.

Al utilizar "joins", podemos cruzar información relevante entre tablas relacionadas y mostrarla en una sola consulta. Por otro lado, las relaciones nos ayudan a mantener la integridad de los datos y garantizar que la información se mantenga consistente a lo largo de diferentes tablas.

En resumen, comprender y dominar los conceptos de "joins" y relaciones en SQL Server es esencial para realizar consultas efectivas y aprovechar al máximo el potencial de las bases de datos relacionales.

Conclusión - Joins y Relaciones

En este módulo dedicado a Joins y Relaciones, hemos comprendido cómo combinar datos de múltiples tablas utilizando los JOINs. Hemos explorado los diferentes tipos de JOINs, como INNER JOIN, LEFT JOIN y RIGHT JOIN, que nos permiten obtener resultados específicos según nuestras

necesidades. Además, hemos comprendido la importancia de las relaciones entre tablas para establecer vínculos entre los datos en SQL Server.

# Subconsultas y Consultas Anidadas

07 | Subconsultas y Consultas Anidadas

En SQL Server, se pueden usar subconsultas y consultas anidadas para realizar consultas más complejas y obtener resultados más específicos. Estas técnicas permiten incluir una consulta dentro de otra consulta, lo que ofrece flexibilidad y control adicional sobre los datos a recuperar.

## Subconsultas

Una subconsulta es una consulta que se incluye dentro de otra consulta. Puede ser utilizada en diferentes cláusulas de una consulta principal, como WHERE o FROM, y se utiliza para filtrar o restringir los resultados de la consulta principal.

Las subconsultas pueden ser muy útiles cuando se necesita buscar datos en función de ciertos criterios o condiciones. Por ejemplo, si queremos obtener todos los empleados que pertenecen a un departamento específico, podemos usar una subconsulta en la cláusula WHERE para buscar los empleados que tienen el mismo identificador de departamento que el departamento objetivo.

Ejemplo de subconsulta en la cláusula WHERE:

```
 SELECT nombre_empleado
FROM empleados
WHERE departamento_id = (SELECT id_departamento
                         FROM departamentos
                         WHERE nombre_departamento = 'Ventas');
```

En este ejemplo, la subconsulta se utiliza para buscar el ID del departamento "Ventas" en la tabla "departamentos", y luego se utiliza ese ID en la consulta principal para recuperar los nombres de los empleados que pertenecen a ese departamento.

## Consultas Anidadas

Las consultas anidadas son consultas que involucran múltiples subconsultas. Son útiles cuando se necesita realizar operaciones más complejas y combinar múltiples condiciones o criterios en una sola consulta.

Por ejemplo, supongamos que queremos obtener el nombre de los empleados que pertenecen a un departamento determinado y tienen un salario superior al promedio de todos los empleados de ese departamento. Podemos utilizar consultas anidadas para lograr esto.

Ejemplo de consulta anidada:

```
 SELECT nombre_empleado
FROM empleados
WHERE departamento_id = (SELECT id_departamento
                         FROM departamentos
                         WHERE nombre_departamento = 'Ventas')
AND salario > (SELECT AVG(salario)
               FROM empleados
               WHERE departamento_id = (SELECT id_departamento
                                        FROM departamentos
                                        WHERE nombre_departamento = 'Ventas'))
```

En este ejemplo, las subconsultas se utilizan para obtener el ID del departamento "Ventas" y el salario promedio de los empleados de ese departamento. Luego, se compara el departamento y el salario en la consulta principal para obtener los nombres de los empleados que cumplen con esas condiciones.

Las subconsultas y las consultas anidadas son herramientas poderosas que permiten realizar consultas más complejas y obtener resultados más específicos en SQL Server. Al dominar estas técnicas, podrás aprovechar al máximo el potencial de este sistema de administración de bases de datos relacional desarrollado por Microsoft.

Conclusión - Subconsultas y Consultas Anidadas

En esta sección sobre Subconsultas y Consultas Anidadas, hemos visto cómo utilizar subconsultas dentro de consultas principales. Estas subconsultas se pueden utilizar en la cláusula WHERE o en la cláusula FROM para filtrar y obtener resultados más complejos. También hemos explorado las consultas anidadas, que involucran múltiples subconsultas, ampliando nuestras posibilidades de obtención de información en SQL Server.

# Ejercicios Practicos

Pongamos en práctica tus conocimientos

08 | Ejercicios Practicos

En esta lección, pondremos la teoría en práctica a través de actividades prácticas. Haga clic en los elementos a continuación para verificar cada ejercicio y desarrollar

habilidades prácticas que lo ayudarán a tener éxito en el tema.

### Instalación de SQL Server

Realiza la instalación de SQL Server en tu equipo siguiendo los pasos proporcionados en la documentación oficial de Microsoft. Al finalizar, verifica que la instalación se haya realizado correctamente y que puedas acceder al SQL Server Management Studio (SSMS).

### Creación de una base de datos

Crea una nueva base de datos llamada 'Tienda' que contenga las siguientes tablas: Clientes, Productos y Pedidos. Define las columnas correspondientes para cada tabla y establece las claves primarias y foráneas necesarias para establecer las relaciones entre ellas.

### Consulta de datos

Realiza las siguientes consultas utilizando la sentencia SELECT: 1. Obtén todos los clientes de la tabla 'Clientes'. 2. Filtra los productos de la tabla

'Productos' que tengan un precio mayor a $50. 3. Ordena los pedidos de la tabla 'Pedidos' por fecha de forma descendente.

Cálculos en conjuntos de datos

Utiliza las funciones de agregación COUNT, SUM, AVG, MIN y MAX para realizar los siguientes cálculos: 1. Obtén el total de clientes en la tabla 'Clientes'. 2. Calcula la suma total del precio de todos los productos en la tabla 'Productos'. 3. Calcula el promedio del precio de los productos en la tabla 'Productos'. 4. Encuentra el precio mínimo y máximo de los productos en la tabla 'Productos'.

Gestión de datos

Realiza las siguientes operaciones de modificación de datos: 1. Inserta un nuevo cliente en la tabla 'Clientes'. 2. Actualiza el precio de un producto específico en la tabla 'Productos'. 3. Elimina un pedido de la tabla 'Pedidos'.

Combina datos de múltiples tablas

Utiliza los diferentes tipos de joins para combinar datos de las tablas 'Clientes', 'Productos' y 'Pedidos': 1. Realiza un INNER JOIN para obtener los clientes que hayan realizado pedidos. 2. Realiza un LEFT JOIN para obtener todos los productos y sus pedidos correspondientes, aunque algunos

productos no tengan pedidos. 3. Realiza un RIGHT JOIN para obtener todos los pedidos y los productos correspondientes, aunque algunos pedidos no tengan productos asociados.

Utilización de subconsultas

Utiliza subconsultas en las siguientes consultas: 1. Filtra los clientes que hayan realizado al menos un pedido utilizando una subconsulta en la cláusula WHERE. 2. Obtiene los productos que tengan un precio mayor al promedio de todos los productos utilizando una subconsulta en la cláusula WHERE. 3. Muestra todos los pedidos junto con la información del cliente que los realizó utilizando una consulta anidada en la cláusula FROM.

# Resumen

Repasemos lo que acabamos de ver hasta ahora

- ✓ En este curso de SQL Server, hemos aprendido los conceptos básicos de esta plataforma de administración de bases de datos. Desde la introducción a SQL Server, pasando por los fundamentos de las bases de datos y las consultas SQL SELECT, hasta llegar a las funciones de agregación y la modificación de datos con INSERT, UPDATE y DELETE. También hemos explorado los JOINs y las relaciones entre tablas, así como las subconsultas y consultas anidadas. Con este conocimiento, los estudiantes estarán preparados para trabajar con eficiencia y solvencia en el manejo de bases de datos en SQL Server.
- ✓ En este módulo de Conceptos Básicos de Bases de Datos, hemos explorado los fundamentos de las bases de datos. Hemos aprendido cómo organizar y almacenar datos en tablas, utilizando filas y columnas. También hemos comprendido la importancia de los tipos de datos para definir la información que se puede almacenar y de las claves primarias y foráneas para establecer relaciones entre tablas. Estos conceptos son esenciales para el correcto diseño y gestión de bases de datos en SQL Server.
- ✓ En esta sección sobre Consultas SQL SELECT, hemos adquirido los conocimientos necesarios para recuperar datos de una tabla utilizando la sentencia SELECT. Hemos aprendido cómo aplicar filtros utilizando la cláusula WHERE y cómo ordenar los resultados con ORDER BY. Estos son fundamentos básicos que nos permitirán realizar consultas eficientes y obtener la información deseada de manera precisa en SQL Server.
- ✓ En este apartado de Funciones de Agregación, hemos explorado las funciones COUNT, SUM, AVG, MIN y MAX, las cuales nos permiten realizar cálculos en conjuntos de datos. Además, hemos comprendido cómo utilizar la cláusula

GROUP BY para agrupar resultados y la cláusula HAVING para filtrar grupos. Estas herramientas son fundamentales para el análisis y la obtención de información resumida en SQL Server.

- ✓ En esta sección de Modificación de Datos, hemos aprendido a insertar nuevos registros con la sentencia INSERT, actualizar datos existentes con la sentencia UPDATE y eliminar registros con la sentencia DELETE. Estas operaciones son fundamentales para mantener la integridad y actualización de los datos en una base de datos en SQL Server.
- ✓ En este módulo dedicado a Joins y Relaciones, hemos comprendido cómo combinar datos de múltiples tablas utilizando los JOINs. Hemos explorado los diferentes tipos de JOINs, como INNER JOIN, LEFT JOIN y RIGHT JOIN, que nos permiten obtener resultados específicos según nuestras necesidades. Además, hemos comprendido la importancia de las relaciones entre tablas para establecer vínculos entre los datos en SQL Server.
- ✓ En esta sección sobre Subconsultas y Consultas Anidadas, hemos visto cómo utilizar subconsultas dentro de consultas principales. Estas subconsultas se pueden utilizar en la cláusula WHERE o en la cláusula FROM para filtrar y obtener resultados más complejos. También hemos explorado las consultas anidadas, que involucran múltiples subconsultas, ampliando nuestras posibilidades de obtención de información en SQL Server.

# Prueba

Comprueba tus conocimientos respondiendo unas preguntas

10 | Prueba

1. ¿Quién desarrolló SQL Server?

- ◯ Oracle
- ◯ Microsoft
- ◯ IBM

2. ¿Qué es una base de datos?

- ◯ Un contenedor organizado para almacenar datos en tablas
- ◯ Un archivo de texto plano
- ◯ Un conjunto de tablas sin relación

3. ¿Qué se utiliza para recuperar datos de una tabla?

- ◯ SELECT
- ◯ INSERT

- ◯ UPDATE

4. ¿Cuál de las siguientes funciones se utiliza para realizar cálculos en conjuntos de datos?

- ◯ COUNT
- ◯ DELETE
- ◯ JOIN

5. ¿Cómo se insertan nuevos registros en una tabla?

- ◯ DELETE
- ◯ UPDATE
- ◯ INSERT

6. ¿Qué tipo de JOIN devuelve todos los registros de la tabla izquierda y sus coincidencias en la derecha?

- ◯ RIGHT JOIN
- ◯ LEFT JOIN
- ◯ INNER JOIN

7. ¿Dónde se pueden utilizar las subconsultas?

- ◯ En la cláusula WHERE
- ◯ En la cláusula ORDER BY

- ◯ En la cláusula GROUP BY

8. ¿Qué se utiliza para eliminar registros de una tabla?

- ◯ INSERT
- ◯ DELETE
- ◯ UPDATE

9. ¿Cuál de las siguientes funciones se utiliza para agrupar resultados basados en una columna?

- ◯ GROUP BY
- ◯ HAVING
- ◯ SELECT

10. ¿Cuál de estas opciones NO es un tipo de JOIN en SQL?

- ◯ INNER JOIN
- ◯ COMBO JOIN
- ◯ OUTER JOIN

11. ¿Qué se utiliza para actualizar datos existentes en una tabla?

- ◯ UPDATE
- ◯ DELETE

- ◯ INSERT

12. ¿Cuál de las siguientes funciones se utiliza para calcular el promedio de valores en una columna?

- ◯ MAX
- ◯ AVG
- ◯ SUM

13. ¿Cuál de estas opciones NO es una consulta anidada?

- ◯ JOIN entre dos tablas
- ◯ SUBQUERY dentro de un FROM
- ◯ SELECT dentro de un SELECT

14. ¿Cuál de las siguientes funciones se utiliza para contar el número de registros en una tabla?

- ◯ COUNT
- ◯ INSERT
- ◯ DELETE

Entregar

Conclusión

# Felicidades!

¡Felicitaciones por completar este curso! Has dado un paso importante para desbloquear todo tu potencial. Completar este curso no se trata solo de adquirir conocimientos; se trata de poner ese conocimiento en práctica y tener un impacto positivo en el mundo que te rodea.

Comparte este curso