



# Índice

<b>0. Cronograma y responsables</b>	<b>2</b>
0.1. Definición del cronograma y planificación de la formación	2
0.2. Responsables formativos y seguimiento	2
<b>1. Introducción</b>	<b>3</b>
1.1. ¿Qué es SQL y por qué es importante?	3
1.2. ¿Qué RDBMS deberíamos aprender?	4
1.3. RDBMS Moderno: Snowflake	5
1.4. Motores de SQL Distribuido: Presto	5
1.4.1. Amazon Athena	6
<b>2. Fundamentos</b>	<b>7</b>
2.1. Tipos de datos	8
2.2. Sentencias SQL	8
2.3. Funciones y operadores DML	9
<b>3. Fase I: Recursos Prácticos en Línea</b>	<b>10</b>
3.1. Udacity: SQL para Análisis de Datos	10
3.2. Tutorial SQLBolt	10
<b>4. Fase II: Declaraciones SQL esenciales</b>	<b>11</b>
<b>5. Fase III: Funciones y Operadores SQL esenciales</b>	<b>13</b>
<b>6. Ejercicios de Evaluación</b>	<b>16</b>
<b>7. Buenas Prácticas y Consideraciones</b>	<b>16</b>
7.1. Consejos	16
7.2. Consejos SQL rápidos en Microsoft SQL Server	18
7.3. Consideraciones	18
7.4. Conceptos clave	20
<b>8. Herramientas y Tecnología</b>	<b>21</b>
<b>9. Figuras</b>	<b>22</b>
Figura 1. Tipos de Join	22
<b>10. Referencias</b>	<b>23</b>



## 0. Cronograma y responsables


### 0.1. Definición del cronograma y planificación de la formación

- La duración estimada del curso es de aproximadamente un mes, lo que equivale a alrededor de 140 horas de estudio

SEMANA DE FORMACIÓN	1	2	3	4		
FASE I						
Fundamentos y Aprendizaje activo						
FASE II						
Instalación Software y Declaraciones SQL con SSM						
FASE III						
Funciones y Operadores Lógicos con SSMS						
EVALUACIÓN FINAL						

### 0.2. Responsables formativos y seguimiento

Nombre	Contacto
Enrique Martín García Martín	enriquemartin.garcia@nter.es
Natalia Hidalgo Gavira	natalia.hidalgo@nter.es

- Asegúrate de estar dentro del espacio de Google '**Formación SQL**'. Esto te permitirá conectarte fácilmente con tus compañeros de curso, plantear preguntas, participar en discusiones y estar al tanto de las reuniones de seguimiento.
- Durante esta primera Fase se espera que registres una línea por día a un documento compartido con el resto de participantes que siga el siguiente formato  Seguimiento Cursantes SQL . Esto nos permitirá tener una visión del progreso individual de cada participante y ayudará a los formadores a identificar las necesidades del grupo que resolveremos sobre la marcha. Además, nos brindará información valiosa para mejorar continuamente el programa de formación.



# 1. Introducción

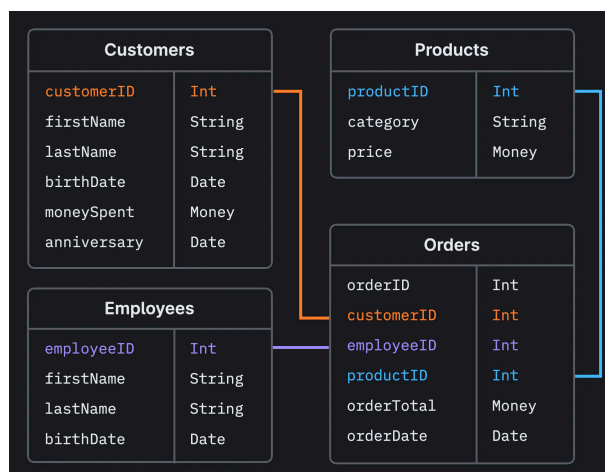
## 1.1. ¿Qué es SQL y por qué es importante?

SQL, o Lenguaje de Consulta Estructurado, es un lenguaje de programación importante en el campo del desarrollo de bases de datos. La mayor diferencia entre SQL y otros lenguajes de programación comunes, como Java y Python, radica en la lógica de diseño del lenguaje. La sintaxis de SQL se centra en la lógica de **agrupamiento**, mientras que la mayoría de los otros lenguajes de programación se centran más en el procedimiento, que utiliza un pensamiento lineal. Aclaración: Hay escenarios en los que puede ser necesario diseñar consultas complejas con un flujo de ejecución más lineal, especialmente al tratar con transformaciones de datos complejas, lógica condicional o múltiples pasos.

Tomemos un breve desvío y veamos un video en YouTube que explica SQL en solo 4 minutos. Este video sirve como una instantánea, una visión de la potencia y eficiencia que SQL aporta. Captura la esencia de SQL, ilustrando su simplicidad, elegancia y el impacto increíble que puede tener en la gestión y extracción de información valiosa de los datos.

[▶ What is SQL? \[in 4 minutes for beginners\]](#)

Este lenguaje está diseñado para permitir a los usuarios hacer consultas, manipular y transformar datos de una base de datos **relacional**. Una base de datos relacional representa una colección de tablas relacionadas (bidimensionales). Cada una de las tablas es similar a una hoja de cálculo de Excel, con un número fijo de columnas con nombres y cualquier cantidad de filas de datos. Aquí tienes una base de datos relacional compuesta por 4 tablas relacionadas con la gestión de ventas y clientes de una empresa de productos:



SQL es una herramienta poderosa para trabajar con datos estructurados en diversos campos, desde administración de bases de datos hasta desarrollo de aplicaciones y análisis de datos. Si disfrutaste del primer video, consulta [▶ How I use SQL as a Data Analyst](#) explicando cómo de útil es SQL para realizar análisis de datos ad hoc y compartir visualizaciones de datos en tiempo real, cuáles son las opciones de SQL más populares y las diferencias entre bases de datos SQL y NoSQL.



## 1.2. ¿Qué RDBMS deberíamos aprender?

Los sistemas de gestión de bases de datos relacionales (RDBMS) están diseñados para administrar y organizar datos en una base de datos relacional. Hay muchos RDBMS populares, incluyendo SQLite, MySQL, PostgreSQL, Oracle y Microsoft SQL Server. Todos ellos admiten el estándar común del lenguaje SQL, pero cada implementación puede diferir en las características adicionales y los tipos de almacenamiento que admite.

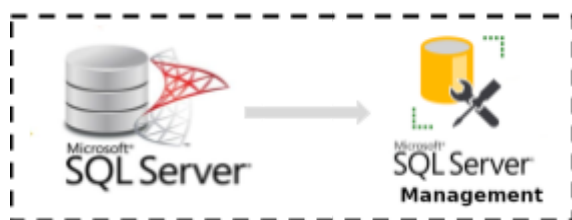
A pesar de estas diferencias, el uso de SQL proporciona un terreno común para trabajar con bases de datos relacionales, y los desarrolladores y administradores de bases de datos a menudo encuentran que sus habilidades en SQL son transferibles entre diferentes sistemas RDBMS, incluso si se necesitan algunos ajustes debido a variaciones en la sintaxis y las características. Cambiar entre bases de datos se convierte en una tarea manejable una vez que comprendes los principios fundamentales de SQL.

A lo largo de esta formación, nos centraremos en construir una base sólida en la sintaxis de SQL. Al entender los conceptos básicos, adquirirás la capacidad de navegar y comandar bases de datos relacionales de manera eficiente. Hasta el año 2024, MySQL, PostgreSQL y Microsoft SQL Server se encuentran entre las tres bases de datos [más populares](#).



- Código abierto:
  - SQLite, MySQL, PostgreSQL
  - Limited versions: Oracle Database Express Edition (XE), Microsoft SQL Server Express, **Microsoft SQL Server Developer Edition**

Este tutorial está diseñado para ofrecer una experiencia de aprendizaje dinámica, incorporando ejercicios interactivos en línea junto con entrenamiento práctico en Microsoft SQL Server Developer Edition. Es una versión gratuita y ampliamente utilizada en la industria, que incluye un potente entorno de desarrollo integrado llamado SQL Server Management Studio (SSMS), proporcionando una interfaz intuitiva para interactuar con el motor de la base de datos SQL Server. Más adelante, en el apartado "[Herramientas y Tecnología](#)", revisaremos el proceso de instalación.





### 1.3. RDBMS Moderno: **Snowflake**

Aunque se adhiere a los principios del modelo relacional, Snowflake introduce varias características innovadoras y diferencias arquitectónicas que lo distinguen de los sistemas tradicionales de RDBMS, tales como:

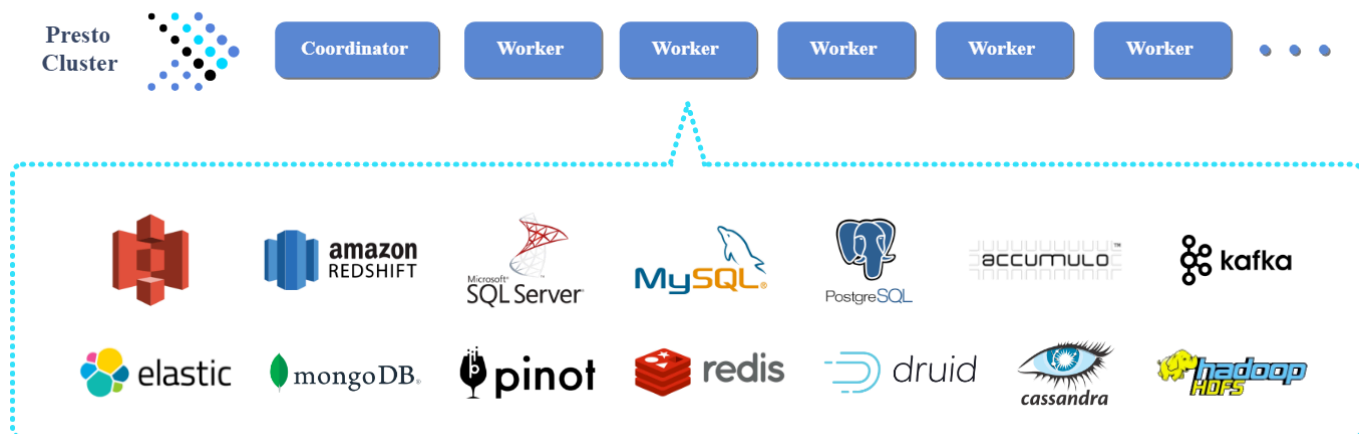
- Diseñado como una solución de almacenamiento de datos nativa de la nube. Está construido para aprovechar la infraestructura en la nube y hacer uso de la escalabilidad y elasticidad ofrecidas por las plataformas en la nube.
- Separación de Almacenamiento y Procesamiento: Uno de los aspectos únicos de Snowflake es su separación de los recursos de almacenamiento y procesamiento. Esta arquitectura permite escalar los recursos de procesamiento de manera independiente del almacenamiento, lo que posibilita un escalado eficiente y flexible desde el punto de vista del costo.
- Compartir Datos de Forma Elástica: Snowflake permite el intercambio elástico de datos, lo que permite a las organizaciones compartir datos de manera segura entre diferentes cuentas de Snowflake. Esto facilita el análisis colaborativo y el intercambio de datos en un entorno multi-nube o multiorganizacional.

El SQL de Snowflake sigue la sintaxis y estructura del SQL estándar, lo que lo hace familiar para los usuarios con experiencia en otros sistemas de gestión de bases de datos relacionales.

### 1.4. Motores de SQL Distribuido: **Presto**

Presto es un motor de SQL distribuido diseñado para ejecutar consultas en big data, es decir, está destinado a manejar procesamiento de datos a gran escala, a menudo en entornos distribuidos o en la nube.

Presto **no es una base de datos** en sí misma, pero permite a los usuarios ejecutar consultas contra cualquier base de datos. Su arquitectura permite a los usuarios realizar consultas en diversas fuentes de datos como Hadoop, AWS S3, Alluxio, MySQL, Cassandra, Kafka y MongoDB. Incluso se pueden realizar consultas en datos de múltiples fuentes en una sola consulta.





### 1.4.1. Amazon Athena

Athena es un servicio de consulta sin servidor de Amazon construido sobre Presto, un motor de SQL distribuido para ejecutar consultas. Esto te permite realizar consultas interactivas en Amazon S3 (Servicio de Almacenamiento en la Nube) utilizando SQL. En general, el objetivo es proporcionar una interfaz SQL familiar para que los usuarios consulten y analicen datos almacenados en Amazon S3.

Cuando interactúas con Athena y escribes consultas SQL, estás utilizando un dialecto SQL que sigue el estándar del lenguaje SQL. Esto significa que muchos de los comandos y declaraciones SQL que uses en Athena serán familiares si tienes experiencia con otras bases de datos relacionales. Ten en cuenta que podrías encontrar algunas diferencias o características específicas que son únicas para Athena o Presto.

- **Nota:** Entre los RDBMS mencionados anteriormente, la sintaxis SQL de Presto es más cercana a la de PostgreSQL.



#### Principales Ventajas

- Sin servidor: No requiere servidor, por lo que el usuario final no necesita preocuparse por la infraestructura, la configuración, el escalado o los fallos. Un analista no necesita gestionar ninguna infraestructura informática subyacente para usarlo.
- No es un servicio de base de datos, por lo tanto, solo pagas por las consultas que ejecutas 💡
  - Más específicamente, el precio es proporcional a la cantidad de datos escaneados por tus consultas.
  - La cantidad de datos almacenados en Amazon S3 (datos de entrada) no afecta directamente los costos de Athena. De manera similar, el volumen de datos que recuperas como resultado de tus consultas no afecta la facturación.
  - El tiempo de ejecución de la consulta no es un factor en la tarificación.
- Implementación Sencilla: Athena no requiere instalación. Se puede acceder directamente desde la Consola de AWS (Interfaz gráfica de usuario basada en web) o AWS CLI (Herramienta de línea de comandos)."

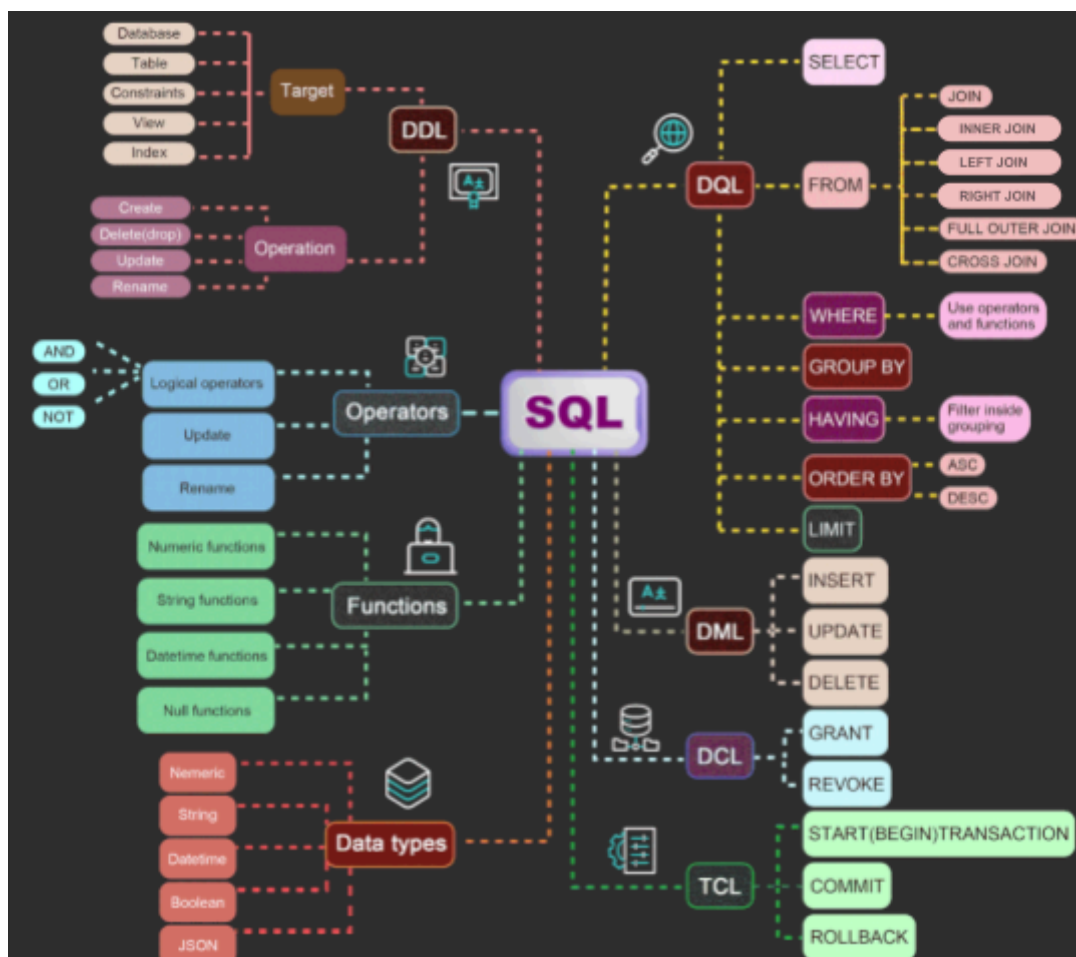


## 2. Fundamentos

Existen muchas fuentes de documentación que pueden ayudarte a aprender los conceptos básicos de SQL. Independientemente de la documentación elegida, se pueden identificar tres temas clave:

- **Tipos de datos:** Esto se refiere a los diferentes tipos de datos que pueden almacenarse en una base de datos, como enteros, cadenas de texto, fechas, etc. Comprender los tipos de datos es crucial para diseñar tablas y garantizar la integridad de los datos.
- **Funciones y operadores de DML:** Esta categoría incluye funciones y operadores utilizados para manipular y procesar datos dentro de una base de datos. Ejemplos incluyen funciones para agregar datos (SUM, AVG, etc.), operadores para filtrar datos (usando la cláusula WHERE) y ordenar datos (con la cláusula ORDER BY).
- **Sentencias SQL:** Las sentencias SQL son comandos que realizan operaciones en los datos, como realizar consultas de información, filtrar datos, ordenar datos, insertar nuevos registros, actualizar registros existentes y eliminar registros. Hay 5 componentes del lenguaje SQL:
  - DQL: Lenguaje de consulta de datos
  - DML: Lenguaje de manipulación de datos
  - DDL: Lenguaje de definición de datos
  - DCL: Lenguaje de control de datos
  - TCL: Lenguaje de control de transacciones

Aquí tienes una visión general de los componentes del lenguaje SQL





Un ingeniero de backend es posible que necesite conocer la mayoría de ellos. Como analista de datos, se hace necesario tener muy buena comprensión de DQL. El énfasis principal de esta formación va a estar en los componentes DQL y DML debido a su amplia aplicabilidad en diferentes perfiles de IT.

En este contexto, hemos compartido documentación relacionada con Athena (basada en Presto, que se asemeja estrechamente a PostgreSQL).

- [Referencia SQL para Athena](#): Amazon Athena admite un subconjunto de sentencias de Lenguaje de Definición de Datos (DDL) y Lenguaje de Manipulación de Datos (DML), así como funciones, operadores y tipos de datos.

Además, la página oficial de Microsoft también incluye una sección bajo el nombre de 'Transact-SQL Reference', el lenguaje SQL que utilizan las aplicaciones y herramientas de Microsoft para conectar e interactuar con los datos almacenados en sus bases de datos. Puede resultar útil cuando practiques habilidades de SQL a través de Microsoft SQL Server Studio:

- [T-SQL: Tipos de datos](#)
- [T-SQL: Funciones](#) y [Operadores](#)
- [T-SQL: Sentencias](#)

## 2.1. Tipos de datos

Cuando se crea una tabla, es necesario especificar los nombres de las columnas y el tipo de datos que cada columna puede contener. Aquí se encuentran los más comúnmente utilizados:

- Boolean
- Integer
- Floating-Point
- Fixed-Precision
- String
- Date and Time
- Structural. Recomendación de lectura: [Consultando arrays](#)
  - Array, Row, Map

## 2.2. Sentencias SQL

### [Declaraciones de Lenguaje de Definición de Datos \(DDL\)](#)

Las declaraciones DDL se utilizan para definir la estructura y organización de la base de datos. Estas declaraciones se ocupan de la creación, modificación y eliminación de objetos de la base de datos. El motor de consultas de Athena se basa en parte en HiveQL DDL. Las declaraciones clave de DDL incluyen:

[CREATE](#): Crea objetos de base de datos como tablas, índices o vistas.

[ALTER](#): Modifica la estructura de los objetos de la base de datos.

[DROP](#): Elimina objetos de la base de datos.





## Declaraciones de Lenguaje de Consulta de Datos (DQL)

El lenguaje de consulta de datos (DQL) en SQL se refiere a la parte del lenguaje SQL que se utiliza para recuperar información de una base de datos. La principal instrucción en DQL es el comando [SELECT](#). Esta instrucción permite seleccionar datos específicos de una o más tablas en una base de datos y mostrarlos en forma de conjunto de resultados.

La sintaxis básica de la instrucción SELECT incluye el nombre de las columnas que se desean seleccionar y la tabla de la cual se quieren recuperar los datos. También puede incluir condiciones adicionales utilizando cláusulas como WHERE para filtrar los resultados, GROUP BY para agrupar los datos, HAVING para aplicar condiciones a grupos de datos y ORDER BY para ordenar los resultados.

En resumen, DQL en SQL se centra en la recuperación y consulta de datos de una base de datos, sin realizar ninguna modificación en los datos almacenados.

## Declaraciones de Lenguaje de Manipulación de Datos (DML)

Las declaraciones DML desempeñan un papel crucial en los procesos de ETL (Extract, Transform, Load), que son flujos de trabajo diseñados para mover y transformar datos desde sistemas fuente hacia sistemas objetivo. Son responsables de dar forma, limpiar y mover datos desde la fuente hasta el objetivo, asegurando que se alinee con la estructura deseada y las reglas comerciales. Las principales operaciones realizadas por las declaraciones DML son:

[INSERT INTO](#): Agrega nuevas filas de datos a una tabla.

[UPDATE](#): Modifica datos existentes en una tabla.

[DELETE](#): Elimina filas de una tabla.

### 2.3. Funciones y operadores DML

Las funciones y operadores a menudo están más directamente alineados con la consulta y análisis de datos, lo que los hace esenciales para personas interesadas en el análisis de datos, informes e inteligencia empresarial.

- Las funciones en la versión 2 del motor de Athena se basan en [Presto 0.217](#) 💡 y esta documentación puede ser de mucha utilidad a la hora de abordar tareas prácticas

Las siguientes fases de formación repasarán las declaraciones, funciones y operadores SQL más relevantes, esenciales para realizar Análisis Exploratorio de Datos (EDA) mediante SQL.



### 3. Fase I: Recursos Prácticos en Línea

Existen numerosos recursos gratuitos de aprendizaje de SQL para enseñar a los principiantes estas nuevas habilidades. Durante la primera semana y media de la formación, se espera la finalización de los dos cursos enumerados. No olvides poner en práctica el contenido de “[Buenas prácticas y Consideraciones](#)”. Se recomienda generar documentos de Google con las respuestas a los ejercicios planteados en cada tutorial, especialmente aquellos en los que te surjan dudas, ayudarán a solventar problemas en grupo y a conocer vuestro seguimiento y progreso:

#### 3.1. [Udacity: SQL para Análisis de Datos](#)

- 3.1.1. Nivel: **Principiante - Intermedio**
- 3.1.2. Modalidad: **Interactive**
- 3.1.3. Lecciones: 7
- 3.1.4. Duración estimada: ~30h
- 3.1.5. **Nota:** No necesitas registrarte en Udacity, puedes acceder con tu cuenta de correo corporativa (o personal) a través de Google.

#### 3.2. [Tutorial SQLBolt](#)

- 3.2.1. Nivel: **Principiante - Intermedio**
- 3.2.2. Lecciones: 20
- 3.2.3. Modalidad: **Interactive**
- 3.2.4. Duración estimada: ~5h

Para reforzar los conocimientos adquiridos en los cursos anteriores, recomendamos continuar con cursos y ejercicios que se listan:

- [LEETCODE: 50 essential SQL questions](#) 💡
  - **Nota:** Se recomienda comenzar estos ejercicios. Muchos de ellos cuentan con solución por lo que puedes autocorregirte. También puedes comparar tu solución con la de otros participantes. Para mantener la consistencia a lo largo de la formación te recomendamos cambiar el RDBMS en el panel `</> Code` a MS SQL Server para usar lenguaje T-SQL. Al final de la formación deberías haber realizado todos los ejercicios.
  - Puedes encontrar ejercicios adicionales en la siguiente [lista](#)
- [SQLZoo Tutorial](#)
  - Nivel: **Principiante - Intermedio**
  - Lecciones: 9
  - Cuestionarios adicionales: 9
  - Modalidad: **Interactive**
  - Duración estimada: ~20h
- [Introduction to SQL: Khan Academy](#)
- [Syllabus: Learn SQL](#)
- [Udacity: Intro to Relational Databases](#)
- [Intro to SQL \(Kaggle\)](#)
- [SQL Desafío Latam](#)



## 4. Fase II: Declaraciones SQL esenciales

Las palabras clave y las cláusulas deben escribirse secuencialmente en las declaraciones SQL. Ejemplo: 'Select', 'from', 'where', 'group by', 'order by', 'limit', y así sucesivamente. Asegúrate de leer la documentación de los siguientes:

1. [SELECT](#)
  - a. WITH
  - b. JOIN\_TYPE: LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN, INNER JOIN
  - c. CROSS JOIN
  - d. GROUP BY
  - e. HAVING
  - f. UNION, INTERSECT, EXCEPT
  - g. ORDER BY
  - h. LIMIT/TOP
  - i. UNNEST
  - j. EXISTS, IN
2. [ANALYZE](#)
3. [DESCRIBE/SHOW COLUMNS](#)

Al concluir esta fase, tu formador te solicitará que entregues un archivo nombrado nombre\_faseII.sql, el cual deberá contener las soluciones que propones para los ejercicios asignados. Para asegurar una revisión eficiente y ordenada de tus respuestas, te pedimos que incluyas comentarios dentro del archivo, claramente identificando cada ejercicio y su respectivo apartado. No olvides poner en práctica el contenido de "[Buenas prácticas y Consideraciones](#)".

💡 : En cada bloque, se recomienda leer todo el material de apoyo asociado antes de abordar los ejercicios del mismo.

💡 : Además de los tutoriales y material de apoyo proporcionado, te invitamos a descubrir el potencial que tiene ChatGPT para resolver dudas y poner ejemplos prácticos de conceptos que no te quedan del todo claro. Aquí puedes ver un [ejemplo](#) de su respuesta cuando le ponemos en contexto y le preguntamos lo siguiente: "Soy estudiante principiante de SQL y no me queda clara la diferencia entre UNION y UNION ALL. ¿Podrías explicarlo y poner un ejercicio práctico para entenderlo mejor?"



Bloque	Descripción	Contenido	Material de apoyo 	Ejercicios SSMS 
1	Condiciones y Restricciones	WITH, SELECT, DISTINCT, WHERE, EXISTS, IN, LIMIT/TOP	- <a href="#">Cláusula SQL TOP, LIMIT, FETCH FIRST o ROWNUM</a>	<a href="#">Bloque 1</a>
2	Tipos de Join	LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN, INNER JOIN, CROSS JOIN	- <a href="#">Tipos de JOINS explicados con ejemplos</a> - <a href="#">Diferencias entre los tipos de JOINS</a> - <a href="#">Figura 1. Tipos de Join</a> - <a href="#">Crea una secuencia de fecha y tiempo</a>	<a href="#">Bloque 2</a>
3	Agrupación y Ordenación	GROUP BY, ORDER BY		<a href="#">Bloque 3</a>
4	Filtrado de grupos	HAVING	<a href="#">Decidir cuándo usar WHERE o HAVING</a>	<a href="#">Bloque 4</a>
5	Combinando resultados	UNION, INTERSECT, EXCEPT		<a href="#">Bloque 5</a>
6	Creación de Vistas		<a href="#">Cómo crear una vista con SQL Server Management Studio</a>	<a href="#">Bloque 6</a>
7	Creación de procedimientos		-  Material Procedimientos ... - <a href="#">Cómo crear un procedimiento almacenado en SSMS</a> - <a href="#">Cómo buscar procedimientos almacenados</a>	<a href="#">Bloque 7</a>
8	Restricciones	Claves primáreas y foráneas, CHECK, UNIQUE	-  Material Claves primarias ... - <a href="#">Restricciones UNIQUE y CHECK - SQL Server   Microsoft Learn</a> - <a href="#">Formas de declarar restricciones comunes</a>	<a href="#">Bloque 8</a>
9	Crear objetos tabla y modificar sus datos	CREATE, INSERT INTO, UPDATE, DELETE	<a href="#">INSERT INTO comando SQL Server</a>	- Algunos Incluidos en B2 - <a href="#">Bloque 9</a>



## 5. Fase III: Funciones y Operadores SQL esenciales

Las siguientes secciones te permitirán adquirir conocimientos y ganar en creatividad mientras solucionas los ejercicios interactivos y problemas reales a los que te enfrentarás.

1. [Operadores Lógicos](#)
  - a. AND, OR, NOT
2. [Funciones y Operadores de Comparación](#)
  - a. BETWEEN
  - b. IS NULL, IS NOT NULL
  - c. IS DISTINCT FROM, IS NOT DISTINCT FROM
  - d. GREATEST, LEAST
  - e. LIKE
  - f. IN, NOT IN
3. [Expresiones Condicionales](#)
  - a. CASE WHEN
  - b. IF
  - c. COALESCE
  - d. NULLIF
4. [Funciones de Conversión](#)
  - a. cast (to date, to string, etc.)
  - b. typeof/data\_type
5. [Funciones y Operadores Matemáticos](#)
  - a. round, truncate
  - b. sign
  - c. abs
6. [Funciones y Operadores String](#)
  - a. concat
  - b. length
  - c. lower, upper
  - d. replace
  - e. reverse
  - f. split, substr
7. [Funciones de Expresiones Regulares](#)
  - a. regexp\_like
8. [Funciones y Operadores de Fecha y Hora](#)
  - a. date, interval
  - b. timestamp AT TIME ZONE
  - c. current\_date, date
  - d. date\_trunc, date\_diff, date\_add
  - e. date\_format, date\_parse
9. [Funciones de Agregación](#)
  - a. array\_agg, histogram
  - b. arbitrary
  - c. count, count\_if
  - d. avg, sum
  - e. min, max, approx\_percentile



10. [Funciones de Ventana](#)

- a. row\_number, rank, dense\_rank (distinction)
- b. lag, lead

11. [Funciones y Operadores Array](#)

- a. array\_distinct
- b. array\_frequency
- c. array\_except
- d. array\_join
- e. array\_min, array\_max, array\_sum, cardinality
- f. array\_sort
- g. filter, transform
  - i. transform + lambda function
- h. contains
- i. reverse, slice

12. [Funciones y Operadores Map](#)

- a. map
- b. cardinality
- c. map\_filter
- d. map\_keys, map\_values

Al concluir esta fase, tu formador te solicitará que entregues un archivo nombrado nombre\_faseIII.sql, el cual deberá contener las soluciones que propones para los ejercicios asignados. Para asegurar una revisión eficiente y ordenada de tus respuestas, te pedimos que incluyas comentarios dentro del archivo, claramente identificando cada ejercicio y su respectivo apartado.

Bloque	Descripción	Contenido	Material de apoyo 	Ejercicios SSMS 
1	- Operadores Lógicos - Funciones y Operadores de Comparación	BETWEEN, IS NULL, IS DISTINCT FROM, GREATEST/LEAST, LIKE, IN, etc.	 WHERE Clause (SQL) - Filt...	<a href="#">Bloque 1</a>
2	- Funciones de conversión - Funciones y Operadores Matemáticos	CONVERT, TRY_CAST, DATA_TYPE, ROUND, SIGN, ABS, etc.	- <a href="#">Vistas de esquema de información del sistema</a>	<a href="#">Bloque 2</a>
3	- Funciones y Operadores de Texto - Expresiones regulares	CONCAT, STRING_SPLIT, SUBSTRING, CHARINDEX, LIKE, REPLACE, etc.	- <a href="#">Uso de STRING_SPLIT</a>	<a href="#">Bloque 3</a>



4	Funciones de Fecha y hora	ISDATE, GETDATE, SYSDATETIME, CONVERT OR CAST, DATEDIFF, MONTH, YEAR, AT TIME ZONE, DATE_ADD	- <a href="#">Tipos de datos de fecha y hora</a>	<a href="#">Bloque 4</a>
5	Expresiones Condicionales	CASE WHEN, IF, COALESCE/ISNULL	- <a href="#">Observaciones de CASE</a> - <a href="#">Observaciones de IF</a> - <a href="#">Diferencias entre ISNULL y COALESCE</a>	<a href="#">Bloque 5</a>
6	- Funciones de Agregación	COUNT, AVG, SUM, MIN, MAX, ARRAY_AGG, MAP_AGG	- <a href="#">COUNT, SUM, AVG, MIN...</a> - <a href="#">System dynamic management views - histogram</a> - <a href="#">ARRAY_AGG (U-SQL)</a> - <a href="#">MAP_AGG (U-SQL)</a>	<a href="#">Bloque 6</a>
7	- Funciones de Ventana	PARTITION BY, ROW_NUMBER, RANK, DENSE_RANK, LAG, LEAD, PERCENT_RANK, AVG, PERCENTILE_DISC	- <a href="#">Material Funciones de V...</a> - <a href="#">La diferencia entre ROW_NUMBER(), RANK() y DENSE_RANK()</a>	<a href="#">Bloque 7</a>
8	- Consulta de Datos Estructurales	- ARRAY, ROW, MAP - Funciones y Operadores - Notación por Punto		



## 6. Ejercicios de Evaluación

- [Ejercicios de Evaluación](#)

## 7. Buenas Prácticas y Consideraciones

### 7.1. Consejos

#### 7.1.1. Acostúmbrate a agregar **comentarios** a tus consultas SQL

Agregar comentarios a las consultas SQL mejora la legibilidad y facilita el mantenimiento y la colaboración entre desarrolladores.

Los comentarios ofrecen contexto adicional y explicaciones que hacen más fácil para cualquier persona, incluyéndote a ti y a otros desarrolladores, comprender el propósito y la lógica detrás de cada parte de la consulta. Aquí tienes un ejemplo basado en la base de datos relacional proporcionada anteriormente:

```
-- Calculating the average order total for each customer in the last quarter
over products of a certain category
SELECT
    customerID,
    AVG(orderTotal) AS averageOrderTotal
FROM
    Orders
WHERE
    orderDate >= '2023-01-01'
    AND orderDate < '2023-04-01'
    -- Filtering orders within the specified date range
    AND productID IN (SELECT productID FROM Products WHERE category =
'Electronics')
    -- Including only orders containing products in the 'Electronics' category
GROUP BY
    customerID;
```

#### 7.1.2. Limita el conjunto de resultados

Recuperar solo un subconjunto de filas utilizando '[TOP](#)' (en el caso de Microsoft SQL Server) o '[LIMIT](#)' (en el caso de Presto y muchas otras bases de datos) puede acelerar significativamente tu proceso de prueba y depuración. Cuando estás tratando con un conjunto de datos grande, utilizar 'TOP'/'LIMIT' u opciones similares te permite recuperar solo un subconjunto de filas, haciendo que tus consultas sean más rápidas durante la prueba. Si estás trabajando en un entorno de producción o con una base de datos compartida, su uso puede ayudar a conservar recursos al evitar la recuperación de un gran número de filas con fines de prueba.





### 7.1.3. Utiliza nombres de tablas y columnas **descriptivos y sugestivos**

Al emplear nombres descriptivos y significativos para tablas, columnas y otros objetos de la base de datos, se mejora la legibilidad del código, facilitando tanto el entendimiento propio como el de otros desarrolladores.

Esto no solo agiliza el mantenimiento y la revisión del código, sino que también reduce la probabilidad de errores al referenciar elementos en consultas. La consistencia en la elección de nombres sugestivos a lo largo del diseño de la base de datos garantiza coherencia y facilita la colaboración con otros. Sugerencias:

- Evita espacios o caracteres especiales en los nombres de tablas y columnas.
- Evita abreviaciones cuando es posible usar palabras completas
- Elige un estilo consistente de mayúsculas y minúsculas (por ejemplo, [snake case](#) o CamelCase) y mantenlo.

```
-- Example using snake_case
CREATE TABLE employee_info (
    employee_id INT,
    first_name VARCHAR,
    last_name VARCHAR
);
```

### 7.1.4. Evita emplear **palabras reservadas**

Evita el uso de palabras reservadas como nombres de tablas o columnas. Si es necesario, utiliza comillas dobles (en Presto), comillas invertidas (en MySQL y otras RDBMS), o corchetes (en SSMS) para escapar palabras reservadas y utilizarlas como identificadores.

- [Palabras clave reservadas \(T-SQL\)](#)
- [Palabras clave reservadas \(Presto\)](#)

```
-- Example with double quotes in Presto
CREATE TABLE "select" (
    "where" VARCHAR,
    "group" VARCHAR,
    "order" INT
);
```

### 7.1.5. Utiliza paréntesis para condiciones lógicas

Al combinar múltiples condiciones en una cláusula WHERE, el uso de paréntesis es crucial para definir el orden de evaluación y garantizar el resultado lógico deseado. Ejemplo:



```
-- Example with parentheses for clarity
SELECT *
FROM Orders
WHERE
    (status = 'Shipped' OR status = 'Delivered')
    AND (orderDate >= '2023-01-01' AND orderDate < '2023-02-01');
```

## 7.2. Consejos SQL rápidos en Microsoft SQL Server

- Puedes encontrar [aquí](#) 15 consejos prácticos en Microsoft SQL Server

## 7.3. Consideraciones

### 7.3.1. Uso de tablas particionadas

La partición implica dividir una tabla grande en segmentos más pequeños y manejables basados en una columna o columnas específicas. Esto puede mejorar significativamente el rendimiento de las consultas, especialmente para operaciones que involucren solo un subconjunto de los datos.

Cuando se utilizan las particiones de manera efectiva, las consultas pueden evitar escanear toda la tabla y enfocarse en las particiones relevantes, lo que conduce a una ejecución de consultas más rápida y a la reducción de los costos de consulta.

La partición es particularmente beneficiosa para operaciones como filtrado basado en rangos, análisis de series temporales y agregaciones en subconjuntos específicos de los datos. La partición por día o mes es común para datos de series temporales. Aquí tienes un ejemplo:

```
-- Example: Creating a partitioned table based on date
CREATE TABLE Sales (
    product_id INT,
    quantity INT,
    revenue DECIMAL
) WITH (
    partitioned_by = ARRAY['sale_date']
);
```

que podría incluir valores como los de la siguiente tabla:

product_id	quantity	revenue	sale_date
101	25	150.50	2022-01-15
102	30	200.75	2022-01-15
103	20	120.30	2022-01-16



104	15	90.25	2022-01-16
105	40	250.80	2022-01-17
106	35	180.50	2022-01-17
107	18	110.75	2022-01-18
108	22	140.20	2022-01-18

### 7.3.2. Compresión de datos y Almacenamiento de Columnas

Combinar la compresión de datos (gzip, snappy, etc.) y el almacenamiento de columnas (parquet, avro, etc.) puede ser una estrategia poderosa para optimizar aún más el almacenamiento de datos y reducir los costos de las consultas.

Los formatos de almacenamiento de columnas almacenan datos columna por columna en lugar de fila por fila, lo que permite una mejor compresión y un procesamiento de consultas más eficiente. Parquet y Avro son formatos de almacenamiento de columnas populares en el ecosistema de big data.

Al aprovechar estas técnicas, puedes minimizar los costos de almacenamiento, reducir la cantidad de datos escaneados durante las consultas y mejorar el rendimiento general del sistema, especialmente en escenarios intensivos en datos y análisis.

### 7.3.3. Creación de vistas

Las vistas te permiten presentar una representación lógica y amigable para el negocio de los datos, abstrayendo los detalles subyacentes del esquema. Te permiten crear una tabla virtual basada en el resultado de una consulta SELECT, y los usuarios pueden interactuar con la vista como si fuera una tabla regular.

Las vistas pueden ser particularmente útiles para crear una vista unificada de datos distribuidos en múltiples tablas o conjuntos de datos, proporcionando una perspectiva consolidada y cohesiva. Los detalles del esquema de las tablas subyacentes quedan ocultos, y los usuarios pueden concentrarse en la estructura simplificada de la vista.

Una buena práctica al nombrar las vistas puede ser usar el prefijo 'vw\_.' Aquí tienes [información](#) sobre cómo crear esas vistas utilizando el SSMS y un ejemplo:

```
-- Example: Creating an Athena view to simplify complex joins
CREATE VIEW vw_orderDetails AS
SELECT
    o.orderID,
    o.orderDate,
    c.firstName as customerName,
    p.productID,
    p.category,
    o.orderTotal
FROM
```



```
Orders o
JOIN Customers c ON o.customerID = c.customerID
JOIN Products p ON o.productID = p.productID;
```

## 7.4. Conceptos clave

### 7.4.1. Diferenciación entre las cláusulas WHERE y HAVING

La cláusula WHERE se utiliza para filtrar filas antes de agrupar, mientras que la cláusula HAVING se emplea para filtrar los resultados ya agrupados. En resumen, la cláusula WHERE **filtra filas** individuales antes de que ocurra cualquier agrupación, la cláusula GROUP BY organiza las filas restantes en **grupos**, y la cláusula HAVING filtra esos grupos según **condiciones agregadas**.

- Cláusula **WHERE**:
  - La cláusula WHERE se utiliza para filtrar filas antes de que ocurra cualquier agrupación.
  - Se aplica en las primeras etapas de la consulta, actuando sobre filas individuales de las tablas de origen.
  - Las condiciones en la cláusula WHERE se evalúan para cada fila de forma independiente.
  - Ayuda a reducir el conjunto de datos eliminando filas que no cumplen con los criterios especificados.

```
SELECT columna1, columna2
FROM tabla
WHERE condición
```

- Cláusula **GROUP BY**:
  - La cláusula GROUP BY se utiliza para agrupar filas según columnas especificadas.
  - Sigue a la cláusula WHERE y se aplica después de filtrar las filas.
  - Se utiliza típicamente en combinación con funciones de agregación (por ejemplo, SUM, COUNT) para realizar operaciones en grupos de filas.
  - El resultado es un conjunto de filas agrupadas basado en las columnas especificadas.

```
SELECT columna1, COUNT(*)
FROM tabla
WHERE condición
GROUP BY columna1
```

- Cláusula **HAVING**:
  - La cláusula HAVING se utiliza para filtrar los resultados de datos agrupados.
  - Viene después de la cláusula GROUP BY y se aplica al conjunto de resultados después de la agrupación.



- Las condiciones en la cláusula HAVING se utilizan para filtrar grupos basados en valores agregados (por ejemplo, COUNT, SUM).
- Ayuda a refinar aún más los resultados de los datos agrupados.

```
SELECT columna1, COUNT(*)  
FROM tabla  
WHERE condición -- filtramos las filas  
GROUP BY columna1 -- agrupamos filas  
HAVING COUNT(*) > 1 -- filtramos el grupo
```

#### 7.4.2. Conciencia de los valores NULL en SQL

En SQL, un valor NULL representa un valor desconocido. Al trabajar con valores NULL en SQL, es fundamental ser consciente de su impacto por ejemplo en:

- Operaciones Lógicas: Ten precaución al usar valores NULL en operaciones lógicas. Sé consciente de las posibles consecuencias y procura prevenir situaciones en las que los valores NULL puedan afectar negativamente tus resultados.
- Declaraciones de Comparación: Cualquier comparación que involucre un NULL resultará en un valor NULL. Comprende las implicaciones de los NULL en tus declaraciones de comparación y maneja adecuadamente estas situaciones en tus consultas.

Puedes encontrar un documento más extenso con mejores prácticas para escribir queries SQL aquí [Best practices for writing SQL queries](#)

## 8. Herramientas y Tecnología

### 8.1. Cómo prepararse para las prácticas

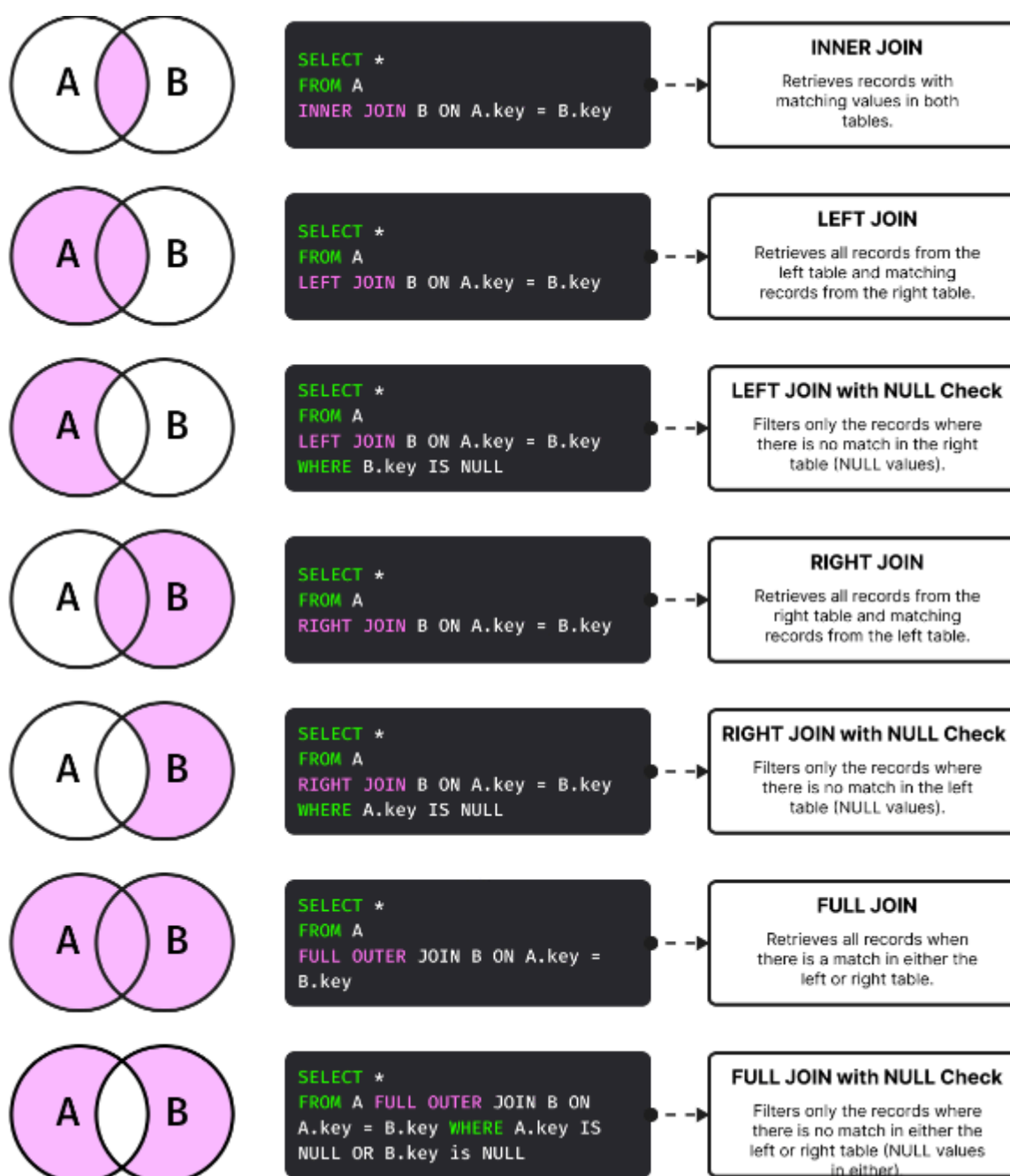
Sigue estos pasos para practicar tus habilidades en tu máquina Windows y comenzar la fase II.

- Descarga datos de muestra adicionales:
  - [SampleData](#)
- Instala Microsoft SQL Server y SQL Server Management Studio (SSMS)
  - [Installing\\_software\\_databases](#)
- Útil: [Cómo cambiar el atajo de teclado para ejecutar comandos SQL](#)



## 9. Figuras

Figura 1. Tipos de Join





## 10. Referencias

- <https://www.bairesdev.com/blog/most-popular-databases/>
- Relacionadas con cursos interactivos:
  - <https://rivery.io/blog/6-best-free-resources-for-learning-sql/>
  - <https://www.classcentral.com/report/best-free-sql-courses/>
  - [Study Plan - LeetCode](#)(Útil para practicar otras tecnologías también)
  - [Free Data Visualization Software | Tableau Public](#)
  - <https://www.devart.com/dbforge/sql/sqlcomplete/sql-join-statements.html>
- <https://medium.com/geekculture/eda-with-sql-mysql-4ac1ea1d977b>
- <https://www.guru99.com/download-install-sql-server.html>
- <https://learn.microsoft.com/es-es/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16>
- <https://dev.to/coderallan/series/17062>
- <https://philip.greenspun.com/sql/index.html>
- Presto
  - <https://prestodb.io/>
  - <https://javahelps.com/presto-sql-for-newbies>
  - <https://prestodb.io/getting-started/>