



PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL
FORMATO GUÍA DE APRENDIZAJE

1. IDENTIFICACIÓN DE LA GUIA DE APRENDIZAJE

- **Denominación del Programa de Formación:** Técnico en programación de software
- **Código del Programa de Formación:** 233104 V2
- **Nombre del Proyecto Formativo (si aplica):** Diseño y desarrollo de soluciones informáticas para las IES del departamento del Tolima articuladas con el centro de comercio y servicios Sena regional Tolima.
- **Fase del Proyecto:** Ejecución
- **Actividad de Proyecto Formativo :** Gestionar la base de datos para el software a desarrollar empleando la normalización de base de datos
- **Competencia:**
220501113: Administrar base de datos de acuerdo con los estándares y requisitos técnicos
- **Resultados:**
220501113-2 Programar sentencias sql en un sistema manejador de bases de datos según requerimientos del cliente
- **Duración de la Guía de Aprendizaje:** 74 Horas



2. PRESENTACIÓN

¡Bienvenidos al Mundo de SQL y las Bases de Datos!

Hola a todos, ¡qué bueno que estén aquí! Hoy vamos a explorar juntos el fascinante mundo de las **bases de datos** y **SQL**.

Pero, ¿por qué deberían importarles estos temas? ¡Porque están en todas partes!

- ¿Alguna vez han usado una app?
- ¿Han jugado algún videojuego online?
- ¿Han buscado algo en Google?



Detrás de todo eso, hay **bases de datos** que guardan información como sus perfiles, resultados, preferencias, y mucho más. **SQL** es el lenguaje que usamos para trabajar con esas bases de datos.

¿Recordemos que es una Base de Datos?

Imagina una **base de datos** como una **gran caja de información** organizada. Dentro de esa caja, todo está bien clasificado: tienes datos de personas, productos, juegos, resultados, etc. ¿Cómo sabemos qué hay dentro de esa caja?

SQL es el lenguaje que usamos para **buscar, agregar, modificar** y **eliminar** esa información.

Piensa en la base de datos como un **armario** y SQL como la llave que te da acceso a todo lo que está allí.



3. FORMULACIÓN DE LAS ACTIVIDADES DE APRENDIZAJE

3.1 Actividades de reflexión inicial:

Descripción de la actividad:

"Tu vida en una base de datos"

Objetivo: Reflexionar sobre el uso de bases de datos en la vida cotidiana y cómo SQL nos permite interactuar con ellas.

Imagina que tu vida diaria está organizada en una base de datos. En esa base de datos, se guardan diferentes tipos de información sobre ti, como tu nombre, tus actividades, tus preferencias, las cosas que compras, etc.

1. Reflexión Personal:

Piensa en varias áreas de tu vida que podrían guardarse en una base de datos. Por ejemplo:



- **Escuela:** ¿Qué datos se guardan sobre ti? ¿Tus notas, tus materias, tus profesores, tus horarios?
- **Redes Sociales:** ¿Qué información hay sobre ti en las redes sociales? ¿Tus amigos, tus fotos, tus publicaciones?
- **Juegos:** Si juegas videojuegos, ¿qué tipo de datos se guardan? ¿Tu puntuación, tu nivel, tus logros?
- **Compras:** Si compras en línea, ¿qué información se guarda? ¿Productos, precios, historial de compras?

2. Preguntas para Reflexionar:

- ¿Cómo crees que se guardan todos estos datos? ¿En qué tipo de "estructura" o "sistema" se organizarían?



- Si tu vida fuera una base de datos, ¿qué tipo de información crees que necesitarías consultar o modificar con más frecuencia? ¿Por ejemplo, ver tu lista de tareas, cambiar tu dirección de correo o revisar tus calificaciones?
- ¿Cómo harías para encontrar rápidamente un dato específico sobre ti, como tu puntuación más alta en un juego o tu promedio en la escuela?
-

3. Actividad en Grupo (opcional):

En parejas o max 3 aprendices vamos a discutir las siguientes preguntas:

- ¿Cómo organizarías la información en una base de datos si fueras a crearla para un videojuego? ¿Qué tablas y campos incluirías?

Ambiente requerido: Sala de informática Institución Educativa Articulada AMT

Estrategias o técnicas didácticas activas: Aprendizaje activo, Aprendizaje colaborativo, Aprendizaje por descubrimiento

Materiales de formación: Portafolio Aprendiz (Plataforma LMS Zajuna) -Computador- Hoja y lapicero- Colores.

Duración de la actividad: 2 horas.

3.2 Actividades de contextualización e identificación de conocimientos necesarios para el aprendizaje:

3.2.1 Fundamentos de Bases de SQL

• ¿Qué es SQL?

SQL (Structured Query Language, o Lenguaje de Consulta Estructurada) es un **lenguaje de programación** utilizado para interactuar con bases de datos. Nos permite hacer cosas como:

- **Consultar** datos (¿quién tiene el promedio más alto en la escuela?).
- **Insertar** datos (agregar un nuevo estudiante).
- **Actualizar** datos (cambiar el promedio de un estudiante).
- **Eliminar** datos (borrar un estudiante de la base de datos).





- **¿Por qué aprender SQL?**

1. Está en todas partes: Como ya mencionamos, aplicaciones, videojuegos, redes sociales y más, usan bases de datos. Aprender SQL te da el poder de crear y gestionar esos sistemas de información.
2. Es útil para el futuro: Cada vez más empresas están buscando personas con habilidades de SQL. Es una de las habilidades más demandadas en el mundo tecnológico.
3. Desarrolla tu lógica y creatividad: SQL no solo es para trabajar con datos, también te hace pensar lógicamente, ¡como un detective de datos!

- **¿Dónde se utiliza SQL?**

SQL (Structured Query Language) se utiliza en casi todos los sistemas que necesitan almacenar y gestionar grandes cantidades de información. Algunos de los lugares y áreas donde SQL es fundamental incluyen:

1. Bases de Datos de Empresas y Organizaciones:

- Las empresas utilizan SQL para gestionar sus bases de datos de clientes, productos, inventarios, ventas y finanzas. Por ejemplo, un supermercado podría usar SQL para llevar el registro de los productos que tiene en stock, las ventas realizadas, los proveedores y la información de los clientes.



2. Redes Sociales:

- Plataformas como Facebook, Instagram, Twitter y TikTok utilizan bases de datos para almacenar toda la información de los usuarios: nombres, publicaciones, fotos, amigos, comentarios y mucho más. SQL permite a estas plataformas realizar consultas y organizar esta información de manera eficiente.



3. Aplicaciones Móviles y Juegos:

- Las aplicaciones que usas en tu teléfono, como las de mensajería, juegos, música o compras, también dependen de SQL para almacenar información sobre tu actividad dentro de la app: el progreso en un juego, las canciones que más escuchas, o el historial de tus compras.

4. Sistemas Bancarios y Financieros:

- Los bancos y otras instituciones financieras usan SQL para gestionar grandes volúmenes de información sobre cuentas bancarias, transacciones, clientes, préstamos y más. Esto les permite realizar consultas rápidas sobre tu saldo, historial de movimientos y otros servicios que ofrecen.



5. E-commerce y Tiendas Online:



- Las tiendas en línea como Amazon o eBay almacenan toda su información sobre productos, precios, clientes y pedidos utilizando bases de datos que se consultan mediante SQL. Cada vez que haces una compra o buscas un producto, la base de datos se consulta para obtener los resultados correctos.

6. Educación:

- Las escuelas y universidades utilizan bases de datos para gestionar los registros de estudiantes, calificaciones, horarios de clases y más. SQL ayuda a organizar y acceder a estos datos de forma rápida y eficiente

AA01 – “El Cuerpo Humano como Base de Datos”

Descripción de la actividad:

Imagina que el cuerpo humano es una base de datos gigante y que cada parte del cuerpo es como una "tabla" que almacena información relacionada. Para que todo funcione de manera correcta, necesitamos organizar esas tablas y establecer **relaciones** entre ellas. ¡Al igual que en una base de datos!

Organización del Cuerpo Humano como Base de Datos: Piensa en las diferentes partes del cuerpo como **tablas**. Algunas de las tablas principales podrían ser:

- **Órganos** (corazón, pulmones, riñones, etc.)
- **Sistema Circulatorio** (arterias, venas, sangre)
- **Sistema Muscular** (músculos, tendones)
- **Sistema Nervioso** (cerebro, nervios)
- **Huesos** (esqueleto, articulaciones)

Crear las tablas:

Piensa en qué información almacenarías en cada tabla. Por ejemplo:

- **Tabla de Órganos:**

ID	Nombre del Órgano	Función Principal	Ubicación
----	-------------------	-------------------	-----------

1	Corazón	Bombear sangre	Pecho
---	---------	----------------	-------



ID Nombre del Órgano Función Principal Ubicación

2	Pulmones	Respirar	Tórax
---	----------	----------	-------

- **Tabla del Sistema Circulatorio:**

ID Tipo de Vaso Sanguíneo Función

1	Arteria	Transportar sangre
2	Vena	Devolver sangre al corazón

b. Relacionar las tablas:

En una base de datos real, las tablas no están separadas; están relacionadas entre sí. Por ejemplo, el **corazón** de la **tabla de Órganos** está relacionado con las **arterias** en la **tabla del Sistema Circulatorio**, porque el corazón bombea sangre a través de las arterias. Podemos establecer esta relación utilizando una clave foránea.

Ambiente requerido: Sala de informática Institución Educativa Articulada AMT

Estrategias o técnicas didácticas activas: Aprendizaje colaborativo, Aprendizaje por descubrimiento

Materiales de formación: Portafolio Aprendiz (Plataforma LMS Zajuna) – Computador - Hoja y lapicero.

Material de apoyo:

Duración de la actividad: 2 horas.

- **Historia breve de SQL**

SQL tiene una historia interesante que se remonta a los primeros días de las bases de datos. Aquí te dejo una breve línea de tiempo:

1. Década de 1970 - Los Primeros Días:

- En los primeros días de la informática, la información se almacenaba de manera muy básica, sin un sistema organizado para consultar o gestionar grandes cantidades de datos. En 1970, un investigador llamado Edgar F. Codd de IBM propuso un modelo nuevo y revolucionario llamado Modelo Relacional de Bases de Datos. Este modelo organizaba la información en tablas (similares a hojas de cálculo) que podían ser relacionadas entre sí.

2. 1974 - Nacimiento de SQL:



- Basado en el modelo relacional, IBM comenzó a desarrollar un lenguaje de consulta para trabajar con esas bases de datos. En 1974, se presentó un lenguaje llamado SEQUEL (Structured English Query Language), que más tarde se convertiría en SQL. SEQUEL fue diseñado para permitir a los usuarios realizar consultas de bases de datos de manera sencilla, usando un lenguaje casi natural.

3. Década de 1980 - SQL Toma Fuerza:

- En los años 80, SQL comenzó a ganar popularidad. En 1986, ANSI (American National Standards Institute) y ISO (International Organization for Standardization) estandarizaron SQL, lo que permitió que el lenguaje fuera adoptado globalmente por diferentes sistemas de bases de datos. Así, SQL pasó de ser una herramienta exclusiva de IBM a convertirse en el estándar para casi todas las bases de datos.

4. 1990 - Expansión y Evolución:

- A medida que las bases de datos y los sistemas informáticos crecían, SQL se fue adaptando y mejorando. Varias compañías comenzaron a desarrollar sus propias versiones de bases de datos SQL, como Oracle, Microsoft SQL Server, MySQL y PostgreSQL.

5. Hoy en Día - SQL como Estándar Global:

- Hoy en día, SQL es el lenguaje más utilizado para gestionar bases de datos. Está presente en prácticamente todos los sistemas que necesitan almacenar y acceder a información. Ya sea para grandes empresas o aplicaciones móviles, SQL sigue siendo la herramienta fundamental para organizar y consultar datos.

AA02 – “ Cuadro Comparativo Bases de Datos”

Descripción de la actividad:

Primero, veremos una pequeña introducción al concepto de **tipos de bases de datos**. Antes que nada debemos entender que no todas las bases de datos son iguales, y cada una tiene características específicas que las hacen más adecuadas para ciertos tipos de aplicaciones o necesidades. Los tipos de bases de datos más comunes incluyen:

- **Bases de datos relacionales (SQL):** Organizan la información en tablas y utilizan SQL para gestionar los datos.



- **Bases de datos no relacionales (NoSQL):** Organizan la información de manera más flexible y pueden usar diferentes estructuras, como documentos, grafos o claves-valor.
- **Bases de datos en la nube:** Son gestionadas en servidores remotos y ofrecen accesibilidad y escalabilidad.
- **Bases de datos orientadas a grafos:** Son ideales para representar relaciones entre elementos (como en redes sociales).
- **Bases de datos orientadas a documentos:** Almacenan datos como documentos JSON o XML.

Distribución de Grupos:

Se debe realizar la siguiente actividad por parejas o si son grupos pequeños se puede asignar a cada grupo un tipo de base de datos para investigar y presentar a la clase. Los grupos deberán investigar el tipo de base de datos asignado y preparar una presentación con la siguiente información:

Para cada tipo de base de datos, deben responder las siguientes preguntas:

- ¿Qué es este tipo de base de datos? (Breve definición)
- ¿Cuáles son sus principales **ventajas**?
- ¿Cuáles son sus principales **desventajas**?
- ¿Qué **casos de uso** o aplicaciones serían ideales para este tipo de base de datos? (Por ejemplo, redes sociales, videojuegos, tiendas online, etc.)
- ¿Cuáles son algunos ejemplos de tecnologías que usan este tipo de base de datos? (Por ejemplo, MySQL para bases de datos relacionales, MongoDB para NoSQL, etc.)

Después de que todos los grupos hayan presentado, vamos a abrir un espacio de discusión. A continuación, respondiendo las siguientes preguntas.

Preguntas de comparación:

- ¿Qué tipo de base de datos sería más adecuado para un sistema de redes sociales como Instagram? ¿Por qué?
- Si tuvieras que crear una tienda online para una empresa pequeña, ¿qué tipo de base de datos elegirías y por qué?
- ¿Qué tipo de base de datos elegirías para una aplicación de mensajería en tiempo real? ¿Por qué?



- ¿Qué tipo de base de datos es más adecuada para manejar grandes volúmenes de datos no estructurados, como imágenes o videos? ¿Y qué tipo para datos más estructurados como registros de clientes?

Tarea (Opcional): El aprendiz debe realizar un cuadro comparativo entre las bases de datos que se socializaron como el siguiente ejemplo:

Ambiente requerido: Sala de informática Institución Educativa Articulada AMT

Estrategias o técnicas didácticas activas: Aprendizaje activo, aprendizaje colaborativo, aprendizaje por descubrimiento

Materiales de formación: Portafolio Aprendiz (Plataforma LMS Zajuna) -Computador- Hoja y lapicero- Colores.

Material de apoyo:

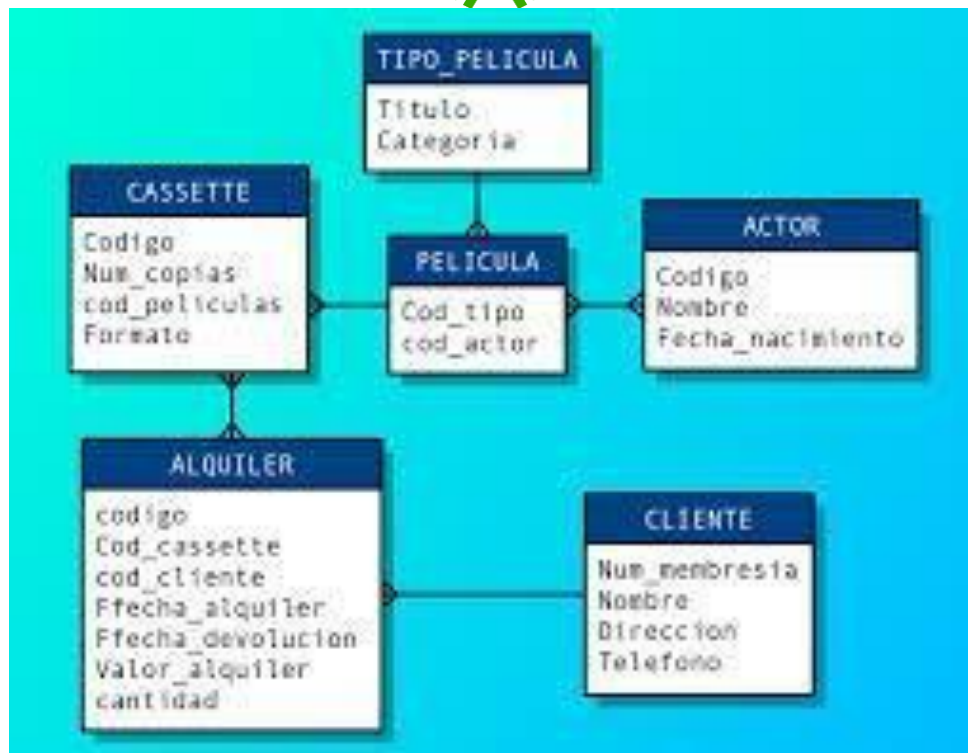
Duración de la actividad: 2 horas.

3.2.2 Conceptos Basicos de SQL

- **Bases de datos y tablas**

Base de datos (DB): Es un conjunto organizado de datos que se almacenan y gestionan electrónicamente. Generalmente se utiliza para almacenar y manipular grandes volúmenes de información de manera eficiente. Las bases de datos pueden almacenar datos de diferentes tipos, como texto, números, imágenes, etc.

Tabla: Es una estructura dentro de una base de datos que organiza los datos en filas y columnas. Cada tabla tiene un nombre único y está compuesta por varios **campos** (columnas) y **registros** (filas). Las tablas permiten representar información relacionada entre sí.



- **Filas y columnas**

Filas (Registros): Son las entradas individuales de una tabla. Cada fila contiene un conjunto de valores que representan un registro o una entidad completa, por ejemplo, una persona, un producto o una venta.

Columnas (Campos): Son las características o atributos de los registros. Cada columna de una tabla tiene un nombre y un tipo de dato específico. Por ejemplo, una columna podría ser "nombre", "edad" o "precio", y cada fila contendría los valores correspondientes a esos atributos.

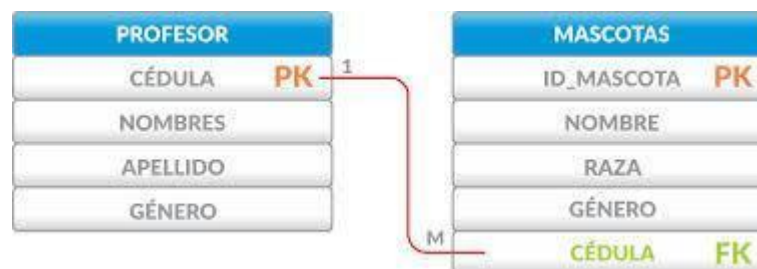


www.codigosql.top

- **Claves primarias y foráneas**

Clave Primaria (Primary Key): Es una columna o conjunto de columnas que identifican de manera única a cada registro en una tabla. No puede haber dos registros con el mismo valor en la clave primaria. La clave primaria es crucial para garantizar la integridad de los datos y evitar duplicados.

Clave Foránea (Foreign Key): Es una columna o un conjunto de columnas que se utiliza para establecer una relación entre dos tablas. La clave foránea hace referencia a la clave primaria de otra tabla. Este vínculo entre tablas permite que los datos se conecten entre sí de manera eficiente.



- **Tipos de datos en SQL**

SQL (Structured Query Language) es el lenguaje utilizado para interactuar con bases de datos. Los tipos de datos son importantes porque definen qué tipo de valores pueden almacenarse en cada columna de una tabla. Algunos de los tipos más comunes son:

- **Enteros (INT):** Para almacenar números enteros.
- **Cadenas de texto (VARCHAR, TEXT):** Para almacenar texto de longitud variable o fija.



- Fecha y hora (DATE, DATETIME, TIMESTAMP): Para almacenar información sobre fechas y horas.
- Decimal (DECIMAL, FLOAT): Para almacenar números con decimales, como cantidades o precios.
- Booleanos (BOOLEAN): Para almacenar valores de verdadero (TRUE) o falso (FALSE).
- Binarios (BLOB): Para almacenar grandes cantidades de datos binarios, como imágenes o archivos.

Ejemplo:

Tipo	Denominación	Muestra
Entero	INT(N)	INT(25)
Decimal	DECIMAL(N,D)	DECIMAL(15,3)
Booleano	BOOL	BOOL
Fecha	DATE	DATE
Fecha y hora	DATETIME	DATETIME
Fecha y hora automática	TIMESTAMP	TIMESTAMP
Hora	TIME	TIME
Año	YEAR(D)	YEAR(10)
Cadena de longitud fija	CHAR(N)	CHAR(3)
Cadena de longitud variable	VARCHAR(N)	VARCHAR(110)

AA03 – “Asignación de datos SQL”

Descripción de la actividad:

Ejercicio 1: Crear una tabla de productos con diferentes tipos de datos

Enunciado: Crea una tabla llamada productos que contenga los siguientes campos:

- **id_producto:** Un campo de tipo **INT** que se autoincremente y sea la clave primaria.



- **nombre:** Un campo de tipo **VARCHAR(100)** que almacene el nombre del producto.
 - **descripcion:** Un campo de tipo **TEXT** para almacenar una descripción larga del producto.
 - **fecha_lanzamiento:** Un campo de tipo **DATE** para almacenar la fecha en que el producto fue lanzado.
 - **precio:** Un campo de tipo **DECIMAL(10, 2)** para almacenar el precio del producto (con 2 decimales).
 - **stock:** Un campo de tipo **INT** para almacenar la cantidad en stock del producto.
 - **activo:** Un campo de tipo **BOOLEAN** para saber si el producto está activo o no.
-

Ejercicio 2: Crear una tabla de empleados con diferentes tipos de datos

Enunciado: Crea una tabla llamada empleados que contenga los siguientes campos:

- **id_empleado:** Un campo de tipo **INT** que se autoincrementa y sea la clave primaria.
 - **nombre:** Un campo de tipo **VARCHAR(50)** que almacene el nombre del empleado.
 - **fecha_nacimiento:** Un campo de tipo **DATE** para almacenar la fecha de nacimiento del empleado.
 - **salario:** Un campo de tipo **DECIMAL(12, 2)** para almacenar el salario del empleado.
 - **fecha_ingreso:** Un campo de tipo **DATE** para almacenar la fecha de ingreso al trabajo.
 - **bono:** Un campo de tipo **FLOAT** para almacenar un bono adicional (puede tener decimales).
 - **activo:** Un campo de tipo **BOOLEAN** para saber si el empleado está activo o no.
-

Ambiente requerido: Sala de informática Institución Educativa Articulada AMT

Estrategias o técnicas didácticas activas: ABP (Aprendizaje Basado en Problemas)

Materiales de formación: Portafolio Aprendiz (Plataforma LMS Zajuna) -Computador- Hoja y lapicero- Colores.

Material de apoyo:

Evidencia de aprendizaje: Prueba de Desempeño



Instrumento de Evaluación: Prueba Practica

Duración de la actividad: 2 horas.

3.3 Actividades de apropiación:

3.3.1 Primeros Pasos con SQL

¡Vamos paso a paso! Aquí te guiaré sobre cómo trabajar con SQL, instalar MySQL y DB Browser for SQLite, y realizar el ejercicio práctico de crear una base de datos y una tabla de "Estudiantes".

1. Herramientas para trabajar con SQL

Existen varias herramientas que te ayudan a interactuar con bases de datos y escribir consultas SQL. Aquí te menciono algunas que puedes usar dependiendo de tus necesidades y del tipo de base de datos que estés utilizando, también tu ordenador en casa o en la Institución Educativa puede variar.

- **MySQL Workbench:** Ideal para trabajar con bases de datos MySQL.
- **DB Browser for SQLite:** Es una herramienta ligera para trabajar con bases de datos SQLite. Recomendado para PC de bajos recursos.
- **DBeaver:** Una herramienta universal que admite múltiples bases de datos como MySQL, PostgreSQL, SQLite, etc.
- **HeidiSQL:** Herramienta ligera para trabajar con bases de datos MySQL/MariaDB.
- **SQL Server Management Studio (SSMS):** Para bases de datos de SQL Server.
- **Visual Studio Code con extensión SQL Server:** Ligero y flexible, puedes escribir SQL con extensiones adecuadas.

2. Instalación de MySQL con DB Browser for SQLite



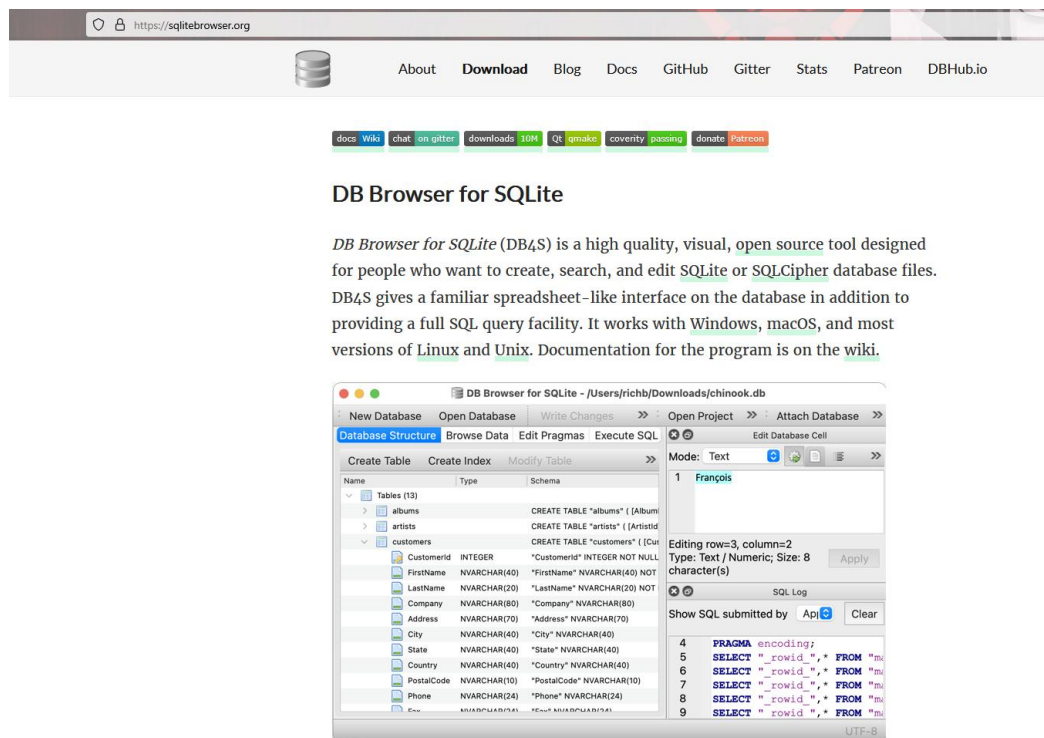
Es importante mencionar que tanto MySQL como DB Browser for SQLite, son dos tecnologías diferentes. MySQL es un sistema de gestión de bases de datos más robusto, mientras que SQLite es una base de datos más ligera y adecuada para entornos locales y con recursos limitados.

Vamos a cubrir la instalación de DB Browser for SQLite ya que es más ligero y adecuado para tu entornos donde estamos limitados, . Si en el futuro necesitas trabajar con MySQL, puedes instalar MySQL y usar MySQL Workbench o DBeaver.

Instalación de DB Browser for SQLite

1. Descargar DB Browser for SQLite:

- Visita la página oficial: [DB Browser for SQLite](https://sqlitebrowser.org)



- Descarga la versión adecuada para tu sistema operativo.

[About](#)[Download](#)[Blog](#)[Docs](#)[GitHub](#)[Gitter](#)[Stats](#)[Patreon](#)[DBHub.io](#)

Downloads

([Please consider sponsoring us on Patreon](#) 😊)

Windows

Our latest release (3.13.1) for Windows:

- [DB Browser for SQLite - Standard installer for 32-bit Windows](#)
- [DB Browser for SQLite - .zip \(no installer\) for 32-bit Windows](#)
- [DB Browser for SQLite - Standard installer for 64-bit Windows](#)
- [DB Browser for SQLite - .zip \(no installer\) for 64-bit Windows](#)

Free code signing provided by [SignPath.io](#), certificate by [SignPath Foundation](#).

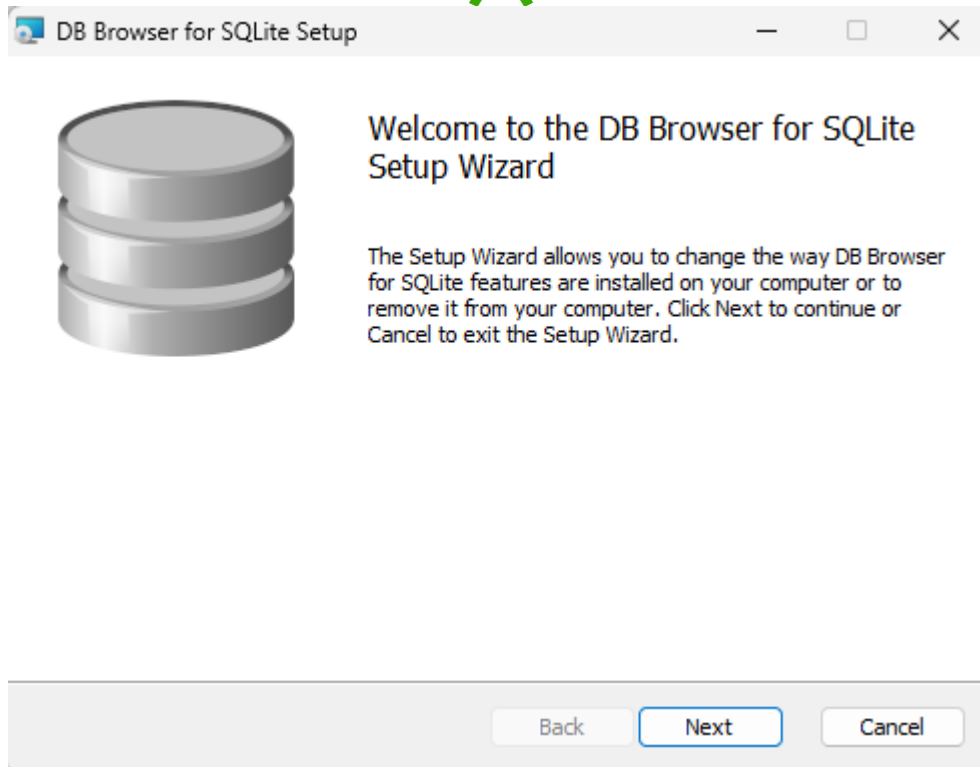
Windows PortableApp

There is a PortableApp available, but it's still the previous (3.12.2) release version. It should be updated to 3.13.1 over the next few days:

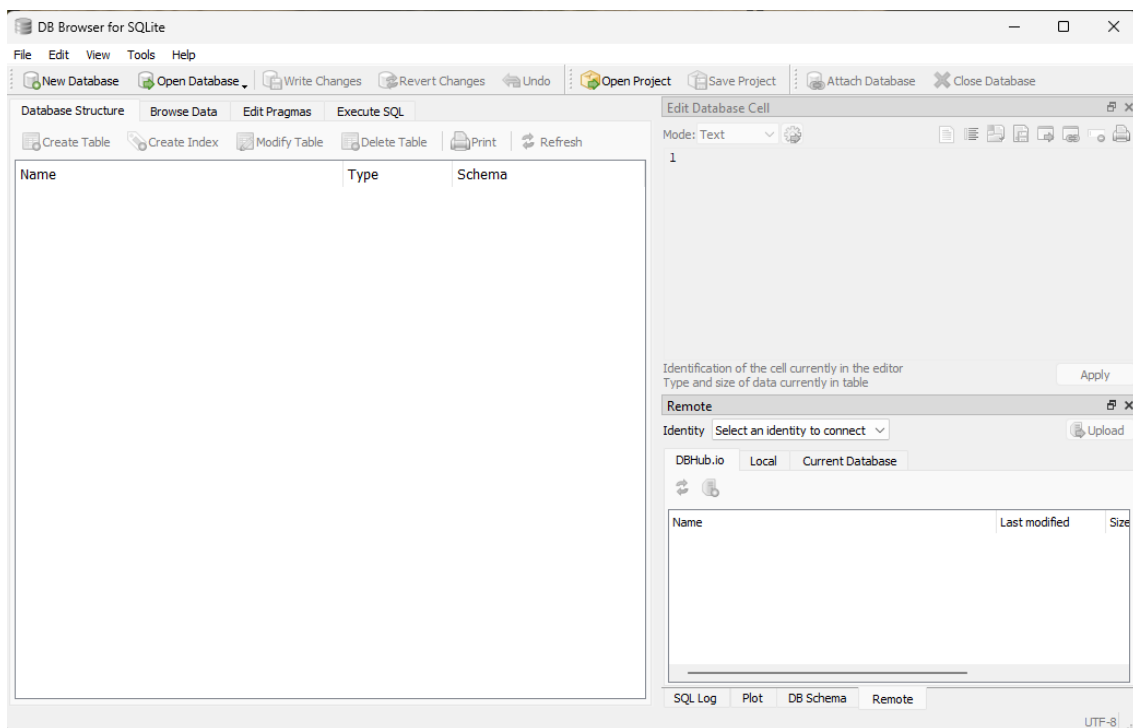
- [DB Browser for SQLite - PortableApp](#)

2. Instalar DB Browser for SQLite:

- Ejecuta el archivo descargado y sigue las instrucciones de instalación.



- Abre DB Browser for SQLite una vez instalado.





Instalación de MySQL (si lo necesitas)

Si más adelante necesitas instalar MySQL para trabajar con bases de datos más grandes, sigue estos pasos:

1. Descargar MySQL:

- Visita [MySQL Downloads](#).

A screenshot of the MySQL Community Downloads page. The page has a light blue header with the text "MySQL Community Downloads" and a circular icon with a plus sign. Below the header, there are two columns of links. The left column includes links to MySQL Yum Repository, MySQL APT Repository, MySQL SUSE Repository, MySQL Community Server, MySQL NDB Cluster, MySQL Router, MySQL Shell, MySQL Operator, MySQL NDB Operator, MySQL Workbench, and MySQL Installer for Windows. The right column includes links to C API (libmysqlclient), Connector/C++, Connector/J, Connector/NET, Connector/Node.js, Connector/ODBC, Connector/Python, MySQL Native Driver for PHP, MySQL Benchmark Tool, Time zone description tables, and Download Archives. On the right side of the page, there is a promotional box for "MySQL Enterprise Edition for Developers" with the text "Free for learning, developing, and prototyping." and a "Download Now »" button. At the bottom of the page, there is a footer with the Oracle logo, "© 2025 Oracle", and links for Privacy, Do Not Sell My Info, Terms of Use, Trademark Policy, and Cookie Preferences.

MySQL Community Downloads

- MySQL Yum Repository
- MySQL APT Repository
- MySQL SUSE Repository
- MySQL Community Server
- MySQL NDB Cluster
- MySQL Router
- MySQL Shell
- MySQL Operator
- MySQL NDB Operator
- MySQL Workbench
- MySQL Installer for Windows
- C API (libmysqlclient)
- Connector/C++
- Connector/J
- Connector/NET
- Connector/Node.js
- Connector/ODBC
- Connector/Python
- MySQL Native Driver for PHP
- MySQL Benchmark Tool
- Time zone description tables
- Download Archives

ORACLE © 2025 Oracle

Privacy / Do Not Sell My Info | Terms of Use | Trademark Policy | Cookie Preferences

MySQL Enterprise Edition for Developers

Free for learning, developing, and prototyping.

Download Now »

- Descarga el MySQL Installer para tu sistema operativo.



MySQL Community Downloads

MySQL Installer

General Availability (GA) ReleasesArchives

MySQL Installer 8.0.41

Note: MySQL 8.0 is the final series with MySQL Installer. As of MySQL 8.1, use a MySQL product's MSI or Zip archive for installation. MySQL Server 8.1 and higher also bundle MySQL Configurator, a tool that helps configure MySQL Server.

Select Version:
8.0.41

Select Operating System:
Microsoft Windows

Windows (x86, 32-bit), MSI Installer (mysql-installer-web-community-8.0.41.0.msi)	8.0.41	2.1M	Download
Windows (x86, 32-bit), MSI Installer (mysql-installer-community-8.0.41.0.msi)	8.0.41	352.2M	Download

We suggest that you use the [MD5 checksums](#) and [GnuPG signatures](#) to verify the integrity of the packages you download.

2. Instalar MySQL:

- Ejecuta el instalador y sigue las instrucciones para instalar el servidor MySQL.
- Durante la instalación, puedes elegir instalar MySQL Workbench si lo deseas para administrar tus bases de datos MySQL.

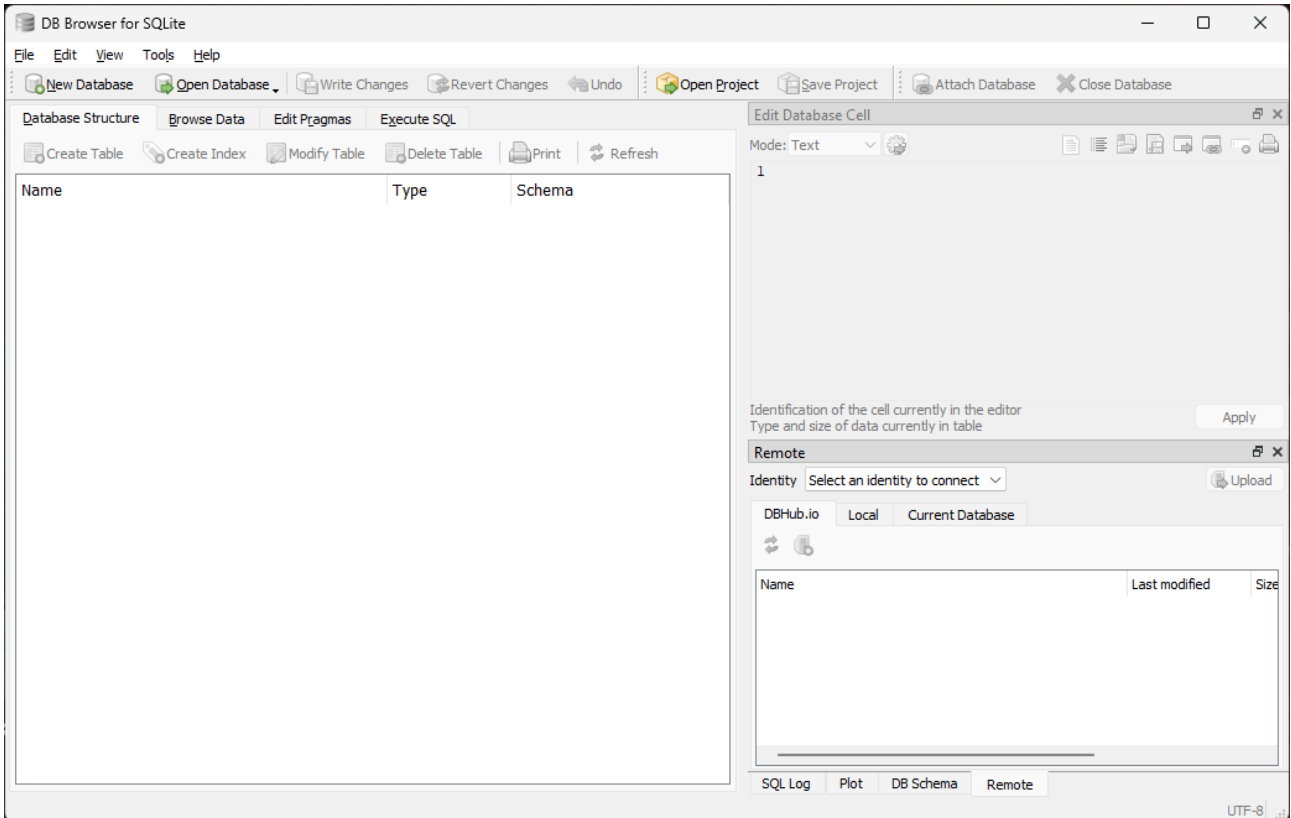
3.3.2. Crear una base de datos

Usando DB Browser for SQLite:

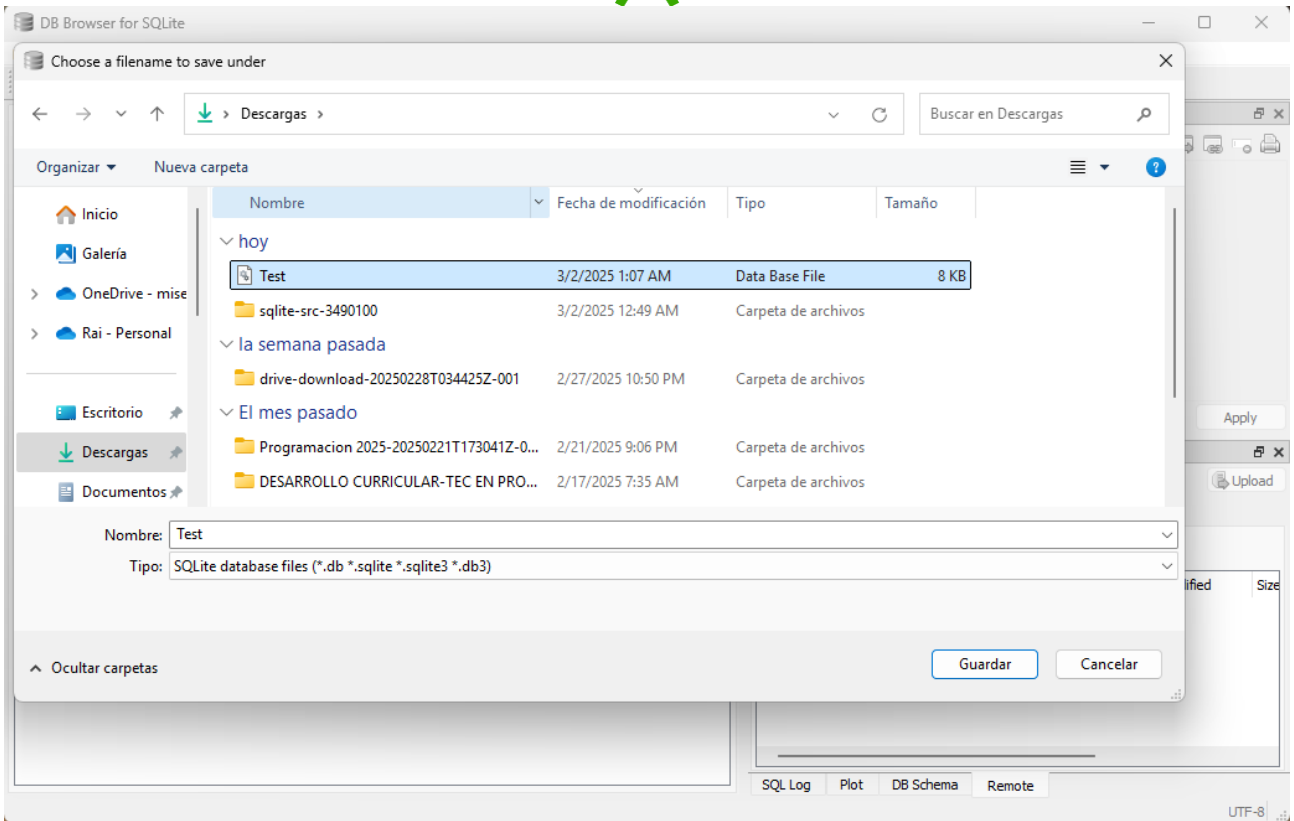
1. Abre DB Browser for SQLite.



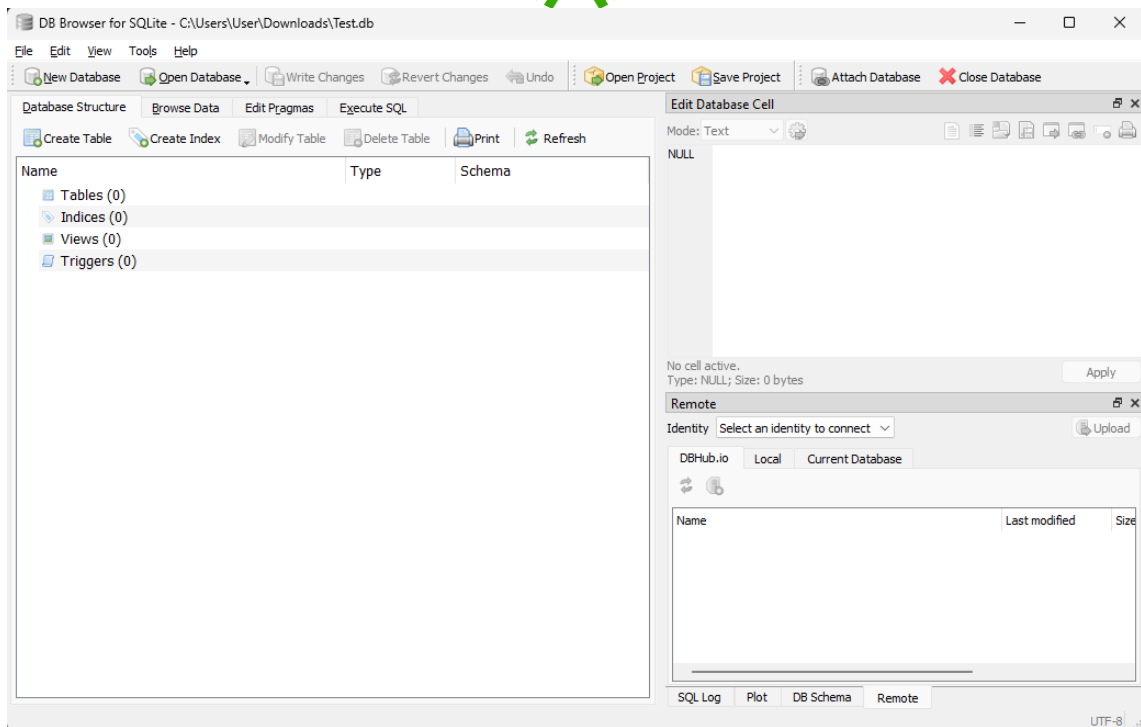
2. Para crear una nueva base de datos, selecciona la opción Nuevo archivo de base de datos en la pantalla de inicio(new database).



3. Elige un nombre y una ubicación para el archivo de base de datos y haz clic en Guardar.



4. Ahora ya tienes tu base de datos creada, y puedes comenzar a agregar tablas.



Usando DB Browser for SQLite:

1. En DB Browser for SQLite, haz clic en Nuevo esquema de tabla.
2. Aparecerá un formulario donde puedes definir el nombre de la tabla y sus columnas.
 - Nombre de la tabla: estudiantes.
 - Columnas:
 - id: Tipo de dato INTEGER (como identificador único, generalmente INTEGER PRIMARY KEY).
 - nombre: Tipo de dato TEXT.
 - edad: Tipo de dato INTEGER.
 - curso: Tipo de dato TEXT.
3. Haz clic en Aceptar para crear la tabla.



AA04 – “ Ejercicios SQL”

Descripción de la actividad:

Ejercicio práctico 1: Crear una base de datos y una tabla de "Estudiantes"

A continuación te guiaremos a través de la creación de una base de datos llamada escuela y una tabla llamada estudiantes.

Usando DB Browser for SQLite:

1. Abre DB Browser for SQLite y crea un nuevo archivo de base de datos, nombrado escuela.db.
2. Crea una nueva tabla llamada estudiantes con las siguientes columnas:
 - id (INTEGER, PRIMARY KEY).
 - nombre (TEXT).
 - edad (INTEGER).
 - curso (TEXT).

Puedes ejecutar también en la consola

CREATE TABLE estudiantes (

id INTEGER PRIMARY KEY AUTOINCREMENT,

nombre TEXT,

edad INTEGER,

curso TEXT

);

3. Haz clic en Ejecutar SQL para crear la tabla.



Ahora vamos a repetir el mismo proceso para las siguientes tablas:

Crea una tabla llamada profesores con las siguientes columnas:

- id: un campo de tipo entero (INT) que se incrementa automáticamente y es la clave primaria de la tabla.
- nombre: un campo de tipo VARCHAR(100) para almacenar el nombre del profesor.
- apellido: un campo de tipo VARCHAR(100) para almacenar el apellido del profesor.
- email: un campo de tipo VARCHAR(100) para almacenar el correo electrónico del profesor.

Crea una tabla llamada materias con las siguientes columnas:

- id: un campo de tipo entero (INT) que se incrementa automáticamente y es la clave primaria de la tabla.
- nombre: un campo de tipo VARCHAR(100) para almacenar el nombre de la materia.
- creditos: un campo de tipo INT para almacenar el número de créditos de la materia.
- id_profesor: un campo de tipo INT que servirá como clave foránea, referenciando el id de la tabla profesores.

Crea una tabla llamada calificaciones con las siguientes columnas:

- id: un campo de tipo entero (INT) que se incrementa automáticamente y es la clave primaria de la tabla.
- id_estudiante: un campo de tipo INT para almacenar el ID del estudiante.



- **id_materia:** un campo de tipo INT que servirá como clave foránea, referenciando el id de la tabla materias.
- **calificacion:** un campo de tipo DECIMAL(5,2) para almacenar la calificación del estudiante en la materia.

Ambiente requerido: Sala de informática Institución Educativa Articulada AMT

Estrategias o técnicas didácticas activas: ABP(Aprendizaje Basado en Problemas)

Materiales de formación: Portafolio Aprendiz (Plataforma LMS Zajuna) -Computador- Hoja y lapicero- Colores.

Material de apoyo:

Evidencia de aprendizaje: Prueba de Desempeño

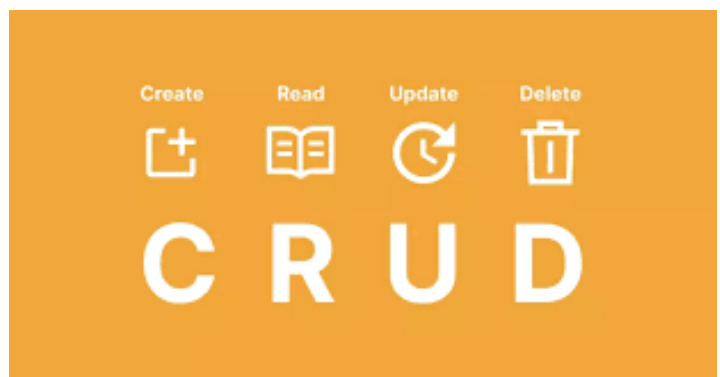
Instrumento de Evaluación: Prueba Practica

Duración de la actividad: 2 horas.

3.4 Actividades de Transferencia el Conocimiento:

3.4.1. Operaciones Básicas SQL

SQL (Structured Query Language) es el lenguaje estándar utilizado para gestionar bases de datos. Con SQL, puedes crear, leer, actualizar y eliminar datos almacenados en bases de datos. Estas operaciones básicas son conocidas como CRUD (Create, Read, Update, Delete), y son esenciales para cualquier persona que desee trabajar con bases de datos.



A continuación, conoceremos las operaciones más comunes que realizarás al interactuar con bases de datos utilizando SQL:



1. Crear (CREATE):

- La operación CREATE se utiliza para crear nuevas bases de datos y tablas. Esta es la primera etapa para estructurar una base de datos en la que se almacenarán los datos. Por ejemplo, podemos crear una base de datos para una tienda o una tabla para almacenar información sobre clientes.

sql

Copy

```
CREATE DATABASE tienda;
CREATE TABLE productos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100),
  precio DECIMAL(10, 2)
);
```

2. Leer (SELECT):

- La operación SELECT permite leer o consultar los datos almacenados en las tablas. Con esta operación, puedes obtener información específica de una o varias tablas de la base de datos. Es la operación más utilizada para ver y analizar los datos.

sql

Copy

```
SELECT * FROM productos;
SELECT nombre, precio FROM productos WHERE precio > 100;
```

3. Actualizar (UPDATE):

- La operación UPDATE se utiliza para modificar los datos existentes en una tabla. Puedes cambiar los valores de una o más columnas de una fila específica, utilizando condiciones para identificar qué registros actualizar.



sql

Copy

```
UPDATE productos  
SET precio = 120  
WHERE id = 1;
```

4. Eliminar (DELETE):

- La operación DELETE permite eliminar datos de una tabla. Puedes borrar registros específicos según una condición, o eliminar todos los registros si no especificas ninguna condición.

sql

Copy

```
DELETE FROM productos WHERE id = 1;
```

Estas cuatro operaciones forman la base de la mayoría de las interacciones que tendrás con bases de datos. Aprender a usar correctamente estas operaciones te permitirá gestionar datos de manera eficiente y efectiva. Además, SQL ofrece muchas otras funcionalidades avanzadas, pero primero es fundamental comprender y dominar estas operaciones básicas.

Con esta base, podrás empezar a gestionar datos en bases de datos, realizar consultas, y hacer modificaciones esenciales en el manejo de la información almacenada.

3.4.2 AVANZADO Y APLICACIONES

Funciones de Agregación

Las **funciones de agregación** en SQL son herramientas que permiten realizar cálculos sobre un conjunto de datos. Estas funciones operan sobre un conjunto de filas y devuelven un solo valor.

Las funciones de agregación más comunes son: COUNT(), SUM(), AVG(), MIN() y MAX().



Aquí te dejo ejemplos de **funciones de agregación** utilizando los mismos temas de las operaciones básicas de SQL que mencionamos antes (como productos, clientes, etc.):

1. COUNT() – Contar el número de registros

La función COUNT() cuenta el número de filas que cumplen con una condición específica. Si se usa sin condiciones, cuenta todas las filas de una tabla o columna.

Ejemplo:

- Contar el número total de productos en la tabla `productos`.

sql

 Copy

```
SELECT COUNT(*) AS total_productos  
FROM productos;
```

- Contar cuántos productos tienen un precio mayor a 100.

sql

 Copy

```
SELECT COUNT(*) AS productos_mayores_a_100  
FROM productos  
WHERE precio > 100;
```

2. SUM() – Sumar valores

La función SUM() calcula la suma de los valores de una columna numérica.



Ejemplo:

- **Suma de los precios de todos los productos en la tabla `productos`.**

sql

Copy

```
SELECT SUM(precio) AS suma_total_precio
FROM productos;
```

- **Suma de los precios de los productos cuyo precio es mayor a 100.**

sql

Copy

```
SELECT SUM(precio) AS suma_precio_mayor_a_100
FROM productos
WHERE precio > 100;
```

3. AVG() – Promedio de valores

La función AVG() calcula el valor promedio de una columna numérica.

Ejemplo:

- **Promedio de los precios de todos los productos.**

sql

Copy

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

- **Promedio de los precios de los productos cuyo precio es mayor a 100.**

sql

Copy

```
SELECT AVG(precio) AS promedio_precio_mayor_a_100
FROM productos
WHERE precio > 100;
```

4. MIN() – Valor mínimo

La función MIN() devuelve el valor más bajo de una columna numérica.



Ejemplo:

- Precio más bajo de todos los productos en la tabla `productos`.

sql

 Copy

```
SELECT MIN(precio) AS precio_minimo
FROM productos;
```

- Precio más bajo de los productos cuyo precio es mayor a 100.

sql

 Copy

```
SELECT MIN(precio) AS precio_minimo_mayor_a_100
FROM productos
WHERE precio > 100;
```

5. MAX() – Valor máximo

La función MAX() devuelve el valor más alto de una columna numérica.

Ejemplo:

- Precio más alto de todos los productos en la tabla `productos`.

sql

 Copy

```
SELECT MAX(precio) AS precio_maximo
FROM productos;
```

- Precio más alto de los productos cuyo precio es mayor a 100.

sql

 Copy

```
SELECT MAX(precio) AS precio_maximo_mayor_a_100
FROM productos
WHERE precio > 100;
```




6. Uso combinado de funciones de agregación con GROUP BY

Las funciones de agregación también se pueden usar junto con GROUP BY para agrupar los datos por una columna y realizar cálculos por cada grupo.

Ejemplo:

- **Promedio de precios por cada fabricante** (suponiendo que la tabla `productos` tenga una columna `id_fabricante` que se relacione con la tabla `fabricantes`).

sql

Copy

```
SELECT id_fabricante, AVG(precio) AS promedio_precio
FROM productos
GROUP BY id_fabricante;
```

- **Número de productos por cada fabricante.**

sql

Copy

```
SELECT id_fabricante, COUNT(*) AS cantidad_productos
FROM productos
GROUP BY id_fabricante;
```

- **Precio más alto y más bajo por cada fabricante.**

sql

Copy

```
SELECT id_fabricante, MAX(precio) AS precio_maximo, MIN(precio) AS precio_minimo
FROM productos
GROUP BY id_fabricante;
```

AA05 – “ Actividad Final”

Descripción de la actividad:

Vamos a realizar un ejercicio completo con todo lo visto anteriormente, el

Ejercicio Completo Final:



1. **Crea una base de datos llamada escuela.**
2. **Usa la base de datos escuela.**
3. **Crea una tabla llamada profesores con las siguientes columnas:**
 - id: un campo de tipo entero (INT) que se incrementa automáticamente y es la clave primaria de la tabla.
 - nombre: un campo de tipo VARCHAR(100) para almacenar el nombre del profesor.
 - apellido: un campo de tipo VARCHAR(100) para almacenar el apellido del profesor.
 - email: un campo de tipo VARCHAR(100) para almacenar el correo electrónico del profesor.
4. **Crea una tabla llamada materias con las siguientes columnas:**
 - id: un campo de tipo entero (INT) que se incrementa automáticamente y es la clave primaria de la tabla.
 - nombre: un campo de tipo VARCHAR(100) para almacenar el nombre de la materia.
 - creditos: un campo de tipo INT para almacenar el número de créditos de la materia.
 - id_profesor: un campo de tipo INT que servirá como clave foránea, referenciando el id de la tabla profesores.
5. **Crea una tabla llamada calificaciones con las siguientes columnas:**
 - id: un campo de tipo entero (INT) que se incrementa automáticamente y es la clave primaria de la tabla.
 - id_estudiante: un campo de tipo INT para almacenar el ID del estudiante.
 - id_materia: un campo de tipo INT que servirá como clave foránea, referenciando el id de la tabla materias.
 - calificacion: un campo de tipo DECIMAL(5,2) para almacenar la calificación del estudiante en la materia.

Ejercicio 2: Insertar Datos en las Tablas

1. Inserta 5 profesores en la tabla profesores
2. Inserta 5 materias en la tabla materias, asignando profesores para cada una de ellas
3. Inserta 5 calificaciones de estudiantes en la tabla calificaciones

Ejercicio 3: Consultas de Datos

1. Consulta todos los profesores y sus respectivas materias.



2. Consulta las calificaciones de un estudiante (por ejemplo, el estudiante con id_estudiante = 1) y las materias correspondientes.
3. Consulta los profesores que enseñan materias con más de 4 créditos.
4. Consulta el promedio de calificaciones de todos los estudiantes por materia.

Ejercicio 4: Actualizar y Eliminar Datos

1. Actualiza la calificación de un estudiante en una materia (por ejemplo, cambia la calificación del estudiante con id_estudiante = 1 en la materia con id_materia = 1 a 9.5)
2. Elimina una materia (por ejemplo, la materia con id = 5)

Ambiente requerido: Sala de informática Institución Educativa Articulada AMT

Estrategias o técnicas didácticas activas: ABP(Aprendizaje Basado en Problemas), Caso de Uso

Materiales de formación: Portafolio Aprendiz (Plataforma LMS Zajuna) -Computador- Hoja y lapicero-Colores.

Material de apoyo:

Evidencia de aprendizaje: Prueba de Desempeño

Instrumento de Evaluación: Prueba Practica

Duración de la actividad: 2 horas.

4. PLANTEAMIENTO DE EVIDENCIAS DE APRENDIZAJE PARA LA EVALUACIÓN EN EL PROCESO FORMATIVO.

Fase del proyecto formativo	Actividad del proyecto formativo	Actividad de Aprendizaje	Evidencias de Aprendizaje	Criterios de Evaluación	Técnicas e Instrumentos de Evaluación
EJECUCION	GESTIONAR LA BASE DE DATOS PARA EL SOFTWARE A DESARROLLAR EMPLEANDO LA	<u>AA01 – “El Cuerpo Humano como Base de Datos”</u>	Preguntas y Respuestas	REALIZA OPERACIONES SOBRE LOS OBJETOS DE LA BASE DE DATOS APLICANDO	Bateria de Preguntas Aprendizaje activo, aprendizaje colaborativo,



	NORMALIZACIÓN DE BASE DE DATOS			LAS INSTRUCCIONES SQL.	aprendizaje por descubrimiento
		<u>AA02 – “Cuadro Comparativo o Bases de Datos”</u>	Preguntas y Respuestas		Bateria de Preguntas Aprendizaje activo, aprendizaje por descubrimiento
		<u>AA03 – “Asignación de datos SQL”</u>	Prueba de Desempeño		Prueba Practica ABP(Aprendizaje Basado en Problemas)
		<u>AA04 – “Ejercicios SQL”</u>	Prueba de Desempeño		Prueba Practica ABP(Aprendizaje Basado en Problemas)
		<u>AA05 – “Actividad Final”</u>	Prueba de Desempeño		Prueba Practica ABP(Aprendizaje Basado en Problemas), Caso de Uso

5. GLOSARIO DE TÉRMINOS

SQL (Structured Query Language): Lenguaje de consulta estructurado utilizado para gestionar y manipular bases de datos relacionales.

Base de Datos (Database): Un conjunto organizado de datos almacenados de manera estructurada que pueden ser accedidos, gestionados y actualizados.

Tabla (Table): Estructura en una base de datos que organiza los datos en filas y columnas. Cada tabla tiene un nombre único dentro de la base de datos.

Fila (Row): También llamada "registro" o "tupla", es una unidad de datos en una tabla. Cada fila contiene datos relacionados entre sí.



Columna (Column): También llamada "campo", es un conjunto de valores de un mismo tipo que se organizan verticalmente en una tabla. Cada columna tiene un nombre y un tipo de datos.

Clave Primaria (Primary Key): Una columna o conjunto de columnas que identifican de manera única cada fila en una tabla. No puede contener valores nulos.

Clave Foránea (Foreign Key): Una columna en una tabla que hace referencia a la clave primaria de otra tabla, creando una relación entre ambas tablas.

Índice (Index): Una estructura que mejora la velocidad de las operaciones de búsqueda en una base de datos. Se crea en una o más columnas de una tabla.

Consulta (Query): Una instrucción SQL escrita para recuperar o manipular datos en una base de datos.

SELECT: Comando SQL utilizado para consultar datos de una tabla. Ejemplo: `SELECT * FROM estudiantes;`

INSERT: Comando SQL utilizado para agregar datos a una tabla. Ejemplo: `INSERT INTO estudiantes (nombre, edad) VALUES ('Juan', 21);`

UPDATE: Comando SQL utilizado para modificar los datos existentes en una tabla. Ejemplo: `UPDATE estudiantes SET edad = 22 WHERE nombre = 'Juan';`

DELETE: Comando SQL utilizado para eliminar datos de una tabla. Ejemplo: `DELETE FROM estudiantes WHERE nombre = 'Juan';`

WHERE: Condición utilizada en una consulta SQL para filtrar los resultados de acuerdo a ciertos criterios. Ejemplo: `SELECT * FROM estudiantes WHERE edad > 18;`

ORDER BY: Comando SQL utilizado para ordenar los resultados de una consulta en función de una o más columnas. Ejemplo: `SELECT * FROM estudiantes ORDER BY edad DESC;`

GROUP BY: Comando SQL utilizado para agrupar los resultados de una consulta según una o más columnas, a menudo junto con funciones agregadas como `COUNT()`, `AVG()`, etc. Ejemplo: `SELECT carrera, COUNT(*) FROM estudiantes GROUP BY carrera;`

JOIN: Operador SQL utilizado para combinar filas de dos o más tablas, basándose en una columna relacionada entre ellas. Ejemplo: `SELECT * FROM estudiantes INNER JOIN cursos ON estudiantes.id = cursos.estudiante_id;`

HAVING: Condición SQL utilizada para filtrar resultados después de haber agrupado los datos con `GROUP BY`. A diferencia de `WHERE`, se aplica a grupos, no a filas individuales. Ejemplo: `SELECT carrera, COUNT(*) FROM estudiantes GROUP BY carrera HAVING COUNT(*) > 5;`



NULL: Un valor especial que representa la ausencia de un valor o dato desconocido. No es lo mismo que un valor vacío o cero.

Subconsulta (Subquery): Una consulta SQL dentro de otra consulta. Se utiliza para realizar una operación en los resultados de la consulta principal.

6. REFERENTES BIBLIOGRÁFICOS

Construya o cite documentos de apoyo para el desarrollo de la guía, según lo establecido en la guía de desarrollo curricular. (**BIBLIOGRAFÍA / WEBGRAFÍA**).

Garcia-Molina, H., Martinez, I., & Valenza, J. (2014). Database systems: The complete book. Prentice Hall.

Ramakrishnan, R., & Gehrke, J. (2014). Database management systems. McGraw-Hill.

Codd, E. F. (1970). A relational model of data for large shared data banks. Communications of the ACM, 13(6), 377-387.

Chen, P. (1976). The entity-relationship model: Toward a unified view of data. ACM Transactions on Database Systems, 1(1), 9-36.

Kent, W. (1983). Database normalization: A step-by-step guide. Database Programming & Design, 6(10), 34-41.

Date, C. (2005). The SQL standard: A review. ACM SIGMOD Record, 34(2), 68-74.

Abadi, D. J. (2009). NoSQL databases: A survey. Proceedings of the VLDB Endowment, 2(2), 1466-1477.

OpenAI. (2023, 18 de noviembre). *ChatGPT* (GPT-4) [Modelo de lenguaje grande]. <https://chat.openai.com>

7. CONTROL DEL DOCUMENTO

	Nombre	Cargo	Dependencia	Fecha
Autor (es)	Jose Luis Bonilla Urrea	Instructor	Tolima C. Comercio y	Febrero 2026



			Servicios	
--	--	--	-----------	--

8. CONTROL DE CAMBIOS (diligenciar únicamente si realiza ajustes a la guía)

	Nombre	Cargo	Dependencia	Fecha	Razón del Cambio
Autor (es)					