**Taller de SQL: Fundamentos y Comandos Avanzados**

***Actividad 1***

**Llave primaria**:

La llave primaria o clave principal es un campo especial que garantiza que cada registro sea único. También se puede definir como una combinación de campos que identifica de forma única cada fila de una tabla y comprende de esta manera una columna o conjunto de columnas.

**Llave foránea**:

Una llave foránea es algo parecido a un vínculo entre dos tablas. Se utiliza para crear una unión con otra tabla la cual es llave primaria. Esta unión o enlace es muy importante porque asegura que los datos en las dos tablas estén relacionados de manera lógica y coherente.

**Relación entre llaves primarias y llaves foráneas:**

Las llaves que mencionamos antes colaboran para conectar tablas en una base de datos. La llave primaria identifica de manera única cada registro en una tabla, mientras que la llave foránea se utiliza en otra tabla para referirse a esa llave primaria y así establece una relación entre ambas.

Para entenderlo mejor lo interpretaba con el ejemplo de que la Llave Primaria seria mi número de cedula ya que es único y no se repite, asegurando que siempre pueda ser identificado. Por otro lado, la Llave Foránea la entiendo suponiendo que hay una lista de cursos en los que estoy inscrito. Cada curso tendría un código que se refiere a mi número de cedula. Ese código es la llave foránea que conecta los cursos con mi información como estudiante.

**Importancia:**

Las llaves primarias y foráneas son esenciales porque ayudan a mantener la base de datos en orden y aseguran que las relaciones entre los datos sean consistentes y validas.

Si no las tuviéramos seria má complicado mantener todo en orden.

***Actividad 2***

SQL significa **Structured Query Language** (Lenguaje de Consulta Estructurado) y es un lenguaje de programación utilizado para trabajar con bases de datos. Básicamente SQL proporciona las instrucciones precisas para comunicarte con dichas o dicha base de datos de manera precisa y efectiva.

**Términos Importantes:**

Consulta:

En SQL, una consulta es una instrucción que le das a la base de datos para obtener información específica.

Tabla:

Una tabla es una estructura que organiza los datos en filas y columnas.

Transacción:

Es un conjunto de operaciones que se ejecutan como una unidad única. Es importante que todas las partes de una transacción se completen con éxito; si una falla, todas las operaciones se revierten para mantener la integridad de los datos.

Relación:

Una relación es una conexión entre dos tablas que se establece a través de llaves primarias y foráneas. Esto permite vincular datos de diferentes tablas de manera lógica.

Columna:

Es un campo en una tabla que almacena un tipo de especifico de datos, como nombres, fechas o cantidades.

Registro:

Un registro es una fila en una tabla que contiene datos sobre una entrada.

**Reflexión**

Saber cómo interactuar con bases de datos usando SQL es fundamental porque en muchas áreas, desde negocios hasta ciencia, necesitamos manejar grandes cantidades de datos. SQL nos permite hacerlo de manera eficiente y precisa.

El término que más me llama la atención es “consulta” porque resume la idea de interactuar directamente con la base de datos lo cual es muy importante para obtener y manipular información relevante de forma ágil.

***Actividad 3***

**Tipos de JOINs en SQL y sus Usos:**

Los JOINs en SQL son herramientas para combinar filas de dos o más tablas basadas en una condición relacionada entre ellas. Entonces viene la explicación de los diferentes tipos de JOINs:

**INNER JOIN**

**·**Descripción: Combina filas de dos tablas cuando hay una coincidencia en ambas tablas.

**·**Uso: Es útil cuando solo quieres las filas que tienen datos correspondientes en ambas tablas.

**LEFT JOIN (LEFT OUTER JOIN)**

**·**Descripción: Devuelve todas las filas de la tabla izquierda y las filas coincidentes de la tabla derecha. Si no hay coincidencia, se devuelve NULL para las columnas de la tabla derecha.

**·**Uso: Es útil cuando quieres todas las filas de la tabla izquierda, independientemente de si tienen coincidencias en la tabla derecha.

**RIGHT JOIN (RIGHT OUTER JOIN)**

**·**Descripción: Devuelve todas las filas de la tabla derecha y las filas coincidentes de la tabla izquierda. Si no hay coincidencia, se devuelve NULL para las columnas de la tabla izquierda.

**·**Uso: Es útil cuando quieres todas las filas de la tabla de la tabla derecha, independientemente de si tiene coincidencias en la tabla izquierda.

**FULL JOIN (FULL OUTER JOIN)**

**·**Descripción: Devuelve todas las filas cuando hay una coincidencia en una de las tablas. Devuelve NULL para las filas de la tabla sin coincidencias.

**·**Uso: Es útil cuando quieres todas las filas de ambas tablas y no quieres perder ninguna información.

| **Tipo de JOIN** | **Filas de Tabla Izquierda (sin coincidencia)** | **Filas de Tabla Derecha (sin coincidencia)** | **Filas con Coincidencia** |
| --- | --- | --- | --- |
| INNER JOIN | No | No | Sí |
| LEFT JOIN | Sí | No | Sí |
| RIGHT JOIN | No | Sí | Sí |
| FULL JOIN | Sí | Sí | Sí |
|  |  |  |  |

***Actividad 4***

1. **SELECT**: Este comando se usa para seleccionar datos de una base de datos. Es el comando más común y básico en SQL.

**·Uso:** SELECT nombre FROM empleados;

**·Utilidad:** Permite extraer datos específicos de una tabla.

1. **WHERE:** Filtra los resultados de una consulta según una condición especifica.

**·Uso:** SELECT nombre FROM empleados WHERE departamento = 'Ventas';

**·Utilidad:** Ayuda a encontrar datos que cumplen con ciertos criterios.

1. **ORDER BY:** Ordena los resultados de una consulta en orden ascendente o descendente.

**·Uso:** SELECT nombre FROM empleados ORDER BY salario DESC;

**·Utilidad:** Facilita la organización de datos para su análisis.

1. **GROUP BY:** Agrupa filas que tienen los mismos valores en una o más columnas.

**·Uso:** SELECT departamento, COUNT(\*) FROM empleados GROUP BY departamento;

**·Utilidad:** Útil para realizar agregaciones y contar registros por categorías.

1. **JOIN:** Combina filas de dos o más tablas basándose en una columna relacionada.

**·Uso:** SELECT \* FROM empleados INNER JOIN departamentos ON empleados.dept\_id = departamentos.id;

**·Utilidad:** Permite relacionar datos de diferentes tablas.

1. **DISTINCT:** Filtra valores duplicados en los resultados de una consulta.

**·Uso:** SELECT DISTINCT nombre FROM empleados;

**·Utilidad:** Asegura que cada valor en el resultado sea único.

1. **LIMIT:** Restringe el número de filas devueltas en una consulta.

**·Uso:** SELECT \* FROM empleados LIMIT 10;

**·Utilidad:** Útil para obtener un subconjunto de datos, especialmente en conjuntos de datos grandes.

1. **CASE:** Proporciona condiciones y permite hacer consultas mas complejas.

**·Uso:** SELECT nombre,

CASE

WHEN salario > 50000 THEN 'Alto'

ELSE 'Bajo'

END AS Salario

FROM empleados;

**·Utilidad:** Permite realizar evaluaciones condiciones dentro de una consulta.