EL RUGIDO DE LOS DATOS JURÁSICOS: EXPLORANDO SQL

ELABORADO POR:

- EDWIN ENRIQUE ALBORNOZ
- JERONIMO ESPITIA JOJOA
- JUAN JOSE OROZCO FERREIRA
- JUAN DIEGO RODRIGUEZ PINEDA
- JUAN DAVID VALENCIA CIFUENTES

¡HOLA! MI NOMBRE ES FAVIO, SOY UN PROFESIONAL EN EL ÁREA DE LA TECNOLOGÍA CON 7 AÑOS DE EXPERIENCIA EN EL ÁREA.

HE TRABAJADO EN DIFERENTES PROYECTOS COMO LA CREACIÓN DE LA NÓMINA DE ATLÉTICO NACIONAL, EN LA BASE DE DATOS DE LA BOLSA DE VALORES DE ESTADOS UNIDOS Y ACTUALMENTE ESTOY TRABAJANDO EN LA EMPRESA ECOPETROL, EN LA ESPECIALIDAD DE CREACIÓN DE BASE DE DATOS.

TAMBIÉN HE EXPLORADO EL DESARROLLO DE LAS BASES DE DATOS CON LOS SOFTWARE SQL WORKBENCH, VISUAL STUDIO Y ACCESS, Y HE TENIDO LA OPORTUNIDAD DE PARTICIPAR EN LA CENTRAL INTELLIGENCE AGENCY (CIA) Y AMAZON EN LA CUAL PROMUEVO LA IMPORTANCIA DE LA INFORMACIÓN DE LOS CLIENTES POR SU SEGURIDAD Y PARA EL CRECIMIENTO DE LAS EMPRESAS SUGIRIENDO RECOMENDACIONES DE SU GUSTO.

EN ESTE CUADERNO DE NOTAS EMPIEZA POR QUE ME GUSTARÍA RECORDAR LOS CONCEPTOS BÁSICOS DE BASE DE DATOS Y LO COMPARTO PARA AYUDAR A MÁS PROGRAMADORES A ENTENDER UN POCO MÁS LAS BASES DE DATOS.

LOS EJEMPLOS DE CÓDIGOS ESTÁN HECHOS EN SQL WORKBENCH.

LAS EMPRESAS SUGIRIENDO RECOMENDACIONES DE SU GUSTO.
EN ESTE CUADERNO DE NOTAS EMPIEZA POR QUE ME GUSTARÍA RECORDAR LOS CONCEPTOS BÁSICOS DE BASE DE DATOS Y LO COMPARTO PARA AYUDAR A MÁS PROGRAMADORES A ENTENDER UN POCO MÁS LAS BASES DE DATOS.
LOS EJEMPLOS DE CÓDIGOS ESTÁN HECHOS EN SQL WORKBENCH.

TABLAS:

Son estructuras de datos que se utilizan para almacenar y organizar información de manera tabular. Son uno de los componentes fundamentales de un sistema de gestión de bases de datos relacionales (RDBMS).

Ejemplo con Dinosaurios

Imagina que los dinosaurio es una tabla en un software especializado en la creación de bases de datos

Los dinosaurios son una tabla pero cada atributo que hay dentro de la tabla es repetitivo a un dinosaurio único (T-Rex, Broncosarios, etc).

Los dinosaurios tienen un número de dientes, nombre, especie, patas y de que se alimenta el dinosaurio que son lo que conforma a la tabla para almacenar cada tipo de dinosaurio.

La tabla tendrá almacenado la información de los diferentes dinosaurios los cuales son datos de su número de dientes, nombre, especie, número de patas y de que se alimenta el dinosaurio.

Ejemplo de código de la tabla dinosaurio:

```sql
CREATE TABLE Dinosaurios (
  ID INT PRIMARY KEY,
  Nombre VARCHAR(50),
  Especie VARCHAR(50),
  Altura FLOAT,
  Peso FLOAT,
  Edad INT
);
```

Tabla: Dinosaurios

CAMPOS:

Se refiere a una columna en una tabla que almacena un dato específico. También se le conoce como atributo. Cada campo tiene un nombre único dentro de una tabla y define el tipo de datos que puede almacenar, como números, cadenas de texto, fechas, etc.

Ejemplo con Dinosaurios

Imagina que una tabla en una base de datos SQL es como una cueva donde los dinosaurios guardan su información. Dentro de esa cueva, hay diferentes compartimentos llamados campos. Cada campo es como un pedazo de arcilla donde los dinosaurios pueden dejar una única pieza de información. Por ejemplo, podríamos tener un campo llamado "Nombre" donde los dinosaurios pueden dejar su nombre, otro campo llamado "Edad" donde dejan su edad, y así sucesivamente.

Ejemplo de código de la Campos dinosaurio:

```
ID INT PRIMARY KEY,
Nombre VARCHAR(50),
Especie VARCHAR(50),
Altura FLOAT,
Peso FLOAT,
Edad INT
```

REGISTROS:

Se refiere a una fila individual en una tabla. Un registro contiene un conjunto completo de valores que corresponden a los atributos o columnas de la tabla.

Ejemplo con Dinosaurios

Imaginemos que cada dinosaurio es un registro en un software especializado en la creación de bases de datos.
Estás en un mundo habitado por diferentes tipos de dinosaurios. Cada dinosaurio es único y tiene sus propias características, como el nombre, la especie, la altura, el peso y la edad. Ahora, supongamos que deseas mantener un registro de todos los dinosaurios que existen en este mundo.
Cada columna en la tabla de la base de datos representa una característica específica de los dinosaurios. Por ejemplo, podrías tener columnas para el nombre, la especie, la altura, el peso y la edad. Estas columnas ayudan a organizar y estructurar los datos de manera que sea fácil buscar, filtrar y realizar cálculos sobre ellos.
Cada registro (fila) en la tabla contendría la información de un dinosaurio específico. Por ejemplo, podrías tener un registro para un dinosaurio llamado "T-Rex" con sus respectivas características en las columnas correspondientes. Otro registro podría ser para un dinosaurio llamado "Triceratops" con sus propias características.

Ejemplo de código de registro dinosaurio:

```sql
INSERT INTO Dinosaurios (ID, Nombre, Especie, Altura, Peso, Edad)
VALUES (1, 'Tyrannosaurus Rex', 'T. Rex', 6.4, 8000, 25);
```

CONSULTAS

Una consulta en MySQL se refiere a una instrucción que se utiliza para recuperar datos de una base de datos. Una consulta en MySQL puede tomar diferentes formas y ser utilizada para diversas operaciones, como seleccionar registros específicos, filtrar datos, realizar cálculos, combinar datos de múltiples tablas, insertar, actualizar o eliminar registros, entre otras cosas.

EJEMPLO CON DINOSAURIOS

imagina que un dinosaurio necesita averiguar una información en específica ya sea los valores de una tabla el dia el mes lo que hará el dinosaurio es buscar una variable correcta para lo que necesite hay diferentes tipos de variables como por ejemplo create table, where, join, from,select, alter,update

WHERE

La cláusula "WHERE" es como un Velociraptor. Este depredador ágil te permite filtrar los resultados de una consulta de la base de datos. Puedes establecer condiciones, como "edad > 18" o "nombre = 'Tyrannosaurus'", para obtener solo los registros que cumplan con esas condiciones.

EJEMPLO CON DINOSAURIO

Supongamos que un dinosaurio tiene una tabla llamada "empleados" con las columnas "id", "nombre", "apellido" y "edad". Y deseas seleccionar solo los empleados cuya edad sea mayor o igual a 30 años.

EJEMPLO EN CÓDIGO EN NUESTRO DINOSISTEMA

```sql
SELECT id, nombre, apellido, edad
FROM empleados
WHERE edad >= 30;
```

Esta consulta seleccionará los registros de la tabla "empleados" donde la columna "edad" sea mayor o igual a 30. Solo se mostrarán las columnas "id", "nombre", "apellido" y "edad" de los empleados que cumplan con esa condición.

Puedes personalizar la consulta según tus necesidades, reemplazando "empleados" por el nombre real de tu tabla y ajustando la condición en la cláusula WHERE según el criterio de filtrado que desees aplicar.

DISTINCT

La cláusula DISTINCT en MySQL se utiliza en una consulta SELECT para eliminar las filas duplicadas y obtener solo los valores únicos de una columna o conjunto de columnas. En otras palabras, esta cláusula permite obtener resultados distintos, sin repetición, basándose en los valores de una o varias columnas específicas.

EJEMPLO CON DINOSAURIO

Supongamos que fabio el dinosaurio tiene una tabla llamada "productos" con las columnas "id", "nombre" y "categoria", y quieres obtener la lista de categorías únicas de los productos.

EJEMPLO EN CÓDIGO EN NUESTRO DINOSISTEMA

```
SELECT DISTINCT categoria FROM productos;
```

Esta consulta devolverá todas las categorías únicas presentes en la tabla "productos" y eliminará cualquier duplicado. Solo se mostrarán las categorías sin repetición.

FROM

La cláusula "FROM" es como un Stegosaurus. Al igual que las placas óseas distintivas en el lomo del Stegosaurus, esta cláusula te permite especificar la tabla o tablas de las que deseas seleccionar los datos. Especificas el nombre de las tablas de las que deseas obtener los datos para realizar tus consultas.

EJEMPLO CON DINOSAURIO

Supongamos que un dinosaurio tiene dos tablas: "empleados" y "departamentos". La tabla "empleados" contiene información sobre los empleados de una empresa, incluyendo sus nombres, apellidos, salarios y el departamento al que pertenecen. La tabla "departamentos" contiene información sobre los diferentes departamentos de la empresa, incluyendo su nombre y ubicación

Deseas obtener una lista de todos los empleados con su nombre, apellido, salario y el nombre del departamento al que pertenecen

EJEMPLO EN CÓDIGO EN NUESTRO DINOSISTEMA

```sql
SELECT empleados.nombre, empleados.apellido, empleados.salario, departamentos.nombre AS nombre_departamento
FROM empleados
JOIN departamentos ON empleados.departamento_id = departamentos.id;
```

La consulta devolverá una lista de empleados con sus nombres, apellidos, salarios y nombres de departamento correspondientes

Recuerda ajustar los nombres de las tablas y las columnas según tu propia estructura de base de datos.

JOIN

es como un Diplodocus. Al igual que este dinosaurio de cuello largo, esta cláusula te permite combinar dos o más tablas en una sola consulta. Puedes unir las tablas en función de una columna común, como un identificador único, y así obtener datos relacionados de diferentes tablas en un solo resultado.

EJEMPLO CON DINOSAURIO

Supongamos que un dinosaurio tiene dos tablas: "clientes" y "pedidos". La tabla "clientes" contiene información sobre los clientes de una tienda, como su ID, nombre y dirección. La tabla "pedidos" contiene información sobre los pedidos realizados por los clientes, como el ID del pedido, el ID del cliente y el total del pedido.
el dino desea obtener una lista de todos los pedidos junto con el nombre y la dirección del cliente que realizó cada pedido

EJEMPLO EN CÓDIGO EN NUESTRO DINOSISTEMA

```sql
SELECT pedidos.id AS id_pedido, clientes.nombre, clientes.direccion, pedidos.total
FROM pedidos
JOIN clientes ON pedidos.cliente_id = clientes.id;
```

La consulta devolverá una lista de pedidos con su ID, total y la información del cliente que realizó cada pedido, incluyendo el nombre y la dirección.
Recuerda ajustar los nombres de las tablas y las columnas según tu propia estructura de base de datos.

SELECT

El "SELECT" es como un Pterodáctilo. Al igual que este dinosaurio volador, esta cláusula te permite seleccionar las columnas específicas que deseas mostrar en los resultados de tu consulta. Puedes elegir las columnas que necesitas para obtener la información deseada.

EJEMPLO CON DINOSAURIO

Supongamos que un dinosaurio tiene una tabla llamada "productos" con las siguientes columnas: "id", "nombre", "precio" y "cantidad".
desea obtener todos los nombres de los productos de la tabla.

EJEMPLO EN CÓDIGO EN NUESTRO DINOSISTEMA

```
SELECT nombre FROM productos;
```

Esta consulta seleccionará la columna "nombre" de la tabla "productos" y devolverá todos los nombres de los productos almacenados en esa tabla.

ALTER

El "ALTER" es como un Anquilosaurio. Al igual que este dinosaurio blindado, esta cláusula te permite realizar cambios en una tabla existente en la base de datos. Puedes agregar, modificar o eliminar columnas de la tabla existente, como una armadura adicional o una espina dorsal modificada.

EJEMPLO CON DINOSAURIO

Supongamos que un dinosaurio tiene una tabla llamada "Usuarios" con las columnas "ID", "Nombre" y "Edad". Si deseas agregar una nueva columna llamada "Correo electrónico" a la tabla existente, puedes usar la cláusula ALTER de la siguiente manera:

EJEMPLO EN CÓDIGO EN NUESTRO DINOSISTEMA

```sql
ALTER TABLE Usuarios
ADD COLUMN `Correo electrónico` VARCHAR(255);
```

Este comando ALTER TABLE agrega una nueva columna llamada "CorreoElectronico" con un tipo de datos VARCHAR y una longitud máxima de 255 caracteres a la tabla "Usuarios". Ahora podrás almacenar direcciones de correo electrónico para cada usuario en tu base de datos.

UPDATE

El "UPDATE" es como un Tyrannosaurus Rex. Al igual que este feroz depredador, esta cláusula te permite actualizar los registros existentes en una tabla. Puedes cambiar los valores de una o más columnas en los registros que cumplan con ciertas condiciones, como modificar la edad de un dinosaurio en particular.

EJEMPLO CON DINOSAURIO

Supongamos que tienes una tabla llamada "Usuarios" con las columnas "ID", "Nombre" y "Edad". Si deseas actualizar el nombre de un usuario en particular en la base de datos, puedes utilizar la cláusula UPDATE de la siguiente manera:

EJEMPLO EN CÓDIGO EN NUESTRO DINOSISTEMA

```
UPDATE Usuarios
SET Nombre = 'NuevoNombre'
WHERE ID = 1;
```

En este ejemplo, la cláusula UPDATE actualiza el valor de la columna "Nombre" a "NuevoNombre" en la tabla "Usuarios" para el registro cuyo ID sea igual a 1. Puedes reemplazar "NuevoNombre" por el valor que desees.

CONSULTA 1

Imaginemos que tenemos una tabla llamada "Pagos_Salarios" que registra los pagos de salarios a los empleados de diferentes sucursales de un parque temático de dinosaurios. Esta tabla tiene columnas como "monto_pago" (que indica la cantidad de dinero pagada) y "fecha_pago" (que registra la fecha en que se realizó el pago). También tenemos otra tabla llamada "Sucursales" que almacena información sobre las diferentes sucursales del parque, incluyendo la columna "nombre_sucursal"

la consulta nos proporciona el total de salarios pagados en el último mes para cada sucursal del parque temático de dinosaurios. Esto nos permite analizar y comparar cuánto se ha pagado a los empleados de cada sucursal en ese período de tiempo determinado.

```sql
SELECT Sucursales.nombre_sucursal AS Sucursal, SUM(Pagos_Salarios.monto_pago) AS TotalSalarios
FROM Pagos_Salarios
JOIN Sucursales ON Pagos_Salarios.sucursal_id = Sucursales.id_sucursal
WHERE Pagos_Salarios.fecha_pago >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
GROUP BY Sucursal;
```

	Sucursal	TotalSalarios
▶	Sucursal A	2500.00
	Sucursal B	3000.00
	Sucursal c	3000.00
	Sucursal d	Sucursal d
	Sucursal e	3000.00

CONSULTA 2

Imaginemos que tenemos un mundo donde existen diferentes empresas que se dedican a la cría y cuidado de dinosaurios. Cada empresa tiene varias sucursales y en cada sucursal trabajan dinoempleados que se encargan de cuidar a los dinosaurios.

El objetivo de esta consulta es obtener información sobre las empresas y la cantidad de dinoempleados que han recibido aumentos salariales en los últimos 6 meses.

Nos brinda el nombre de las empresas y la cantidad de dinoempleados que han recibido aumentos salariales en los últimos 6 meses. Podemos ver cuántas empresas están recompensando a sus dinoempleados y tener una idea del cuidado y bienestar de los dinosaurios en cada empresa.

```sql
SELECT Empresas.nombre_empresa AS Empresa, COUNT(DISTINCT Aumentos_Salariales.empleado_id) AS NumEmpleados
FROM Empresas
LEFT JOIN Sucursales ON Empresas.id_empresa = Sucursales.empresa_id
LEFT JOIN Empleados ON Sucursales.id_sucursal = Empleados.sucursal_id
LEFT JOIN Aumentos_Salariales ON Empleados.id_empleado = Aumentos_Salariales.empleado_id
WHERE Aumentos_Salariales.fecha_aumento >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
GROUP BY Empresa;
```

Empresa	NumEmpleados
arturo calle	1
ecopetrol	1
exito	1
movistar	1
terpel	1

CONSULTA 3

Supongamos que en el mundo de los dinosaurios existen diferentes departamentos encargados de tareas específicas, como el cuidado de los huevos, la alimentación de los dinosaurios, la limpieza de los recintos, etc. Además, cada dinoempleado está asignado a un departamento en particular.

El objetivo de esta consulta es obtener información sobre los departamentos y el número de dinoempleados que han recibido aumentos salariales en los últimos 6 meses.

Nos proporciona el nombre de los departamentos y el número de dinoempleados que han recibido aumentos salariales en los últimos 6 meses. Esto nos permite tener una idea de qué departamentos están recompensando a sus dinoempleados y valorando su trabajo en el mundo de los dinosaurios.

```sql
SELECT D.nombre_departamento, COUNT(A.empleado_id) AS numero_empleados_aumento
FROM Departamentos D
LEFT JOIN Empleados E ON D.id_departamento = E.departamento_id
LEFT JOIN Aumentos_Salariales A ON E.id_empleado = A.empleado_id
WHERE A.fecha_aumento >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
GROUP BY D.id_departamento;
```

nombre_departamento	numero_empleados_aumento
Ventas	1
recursos	1
mantenimiento	1
soporte	1
ingenerios	1

CONSULTA 4

Imaginemos que en el mundo de los dinosaurios existen diferentes sucursales donde se llevan a cabo diversas actividades relacionadas con la crianza y cuidado de los dinosaurios. Cada sucursal tiene empleados encargados de estas tareas.

El objetivo de esta consulta es obtener información sobre las sucursales y el número de dinoempleados que han recibido aumentos salariales en los últimos 6 meses. Nos brinda el nombre de las sucursales y el número de dinoempleados que han recibido aumentos salariales en los últimos 6 meses. Esto nos permite tener una visión general de cuántos dinoempleados están siendo recompensados en cada sucursal del mundo de los dinosaurios.

```sql
SELECT s.nombre_sucursal, COUNT(DISTINCT a.empleado_id) AS numero_empleados_aumento
FROM Aumentos_Salariales a
INNER JOIN Empleados e ON a.empleado_id = e.id_empleado
INNER JOIN Sucursales s ON e.sucursal_id = s.id_sucursal
WHERE a.fecha_aumento >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
GROUP BY s.nombre_sucursal;
```

	nombre_sucursal	numero_empleados_aumento
▶	Sucursal A	1
	Sucursal B	1
	Sucursal c	1
	Sucursal d	1
	Sucursal e	1

CONSULTA 5

Imaginemos que en el mundo de los dinosaurios existen diferentes empresas que se encargan de la cría y cuidado de dinosaurios. Cada empresa tiene varias sucursales y en cada sucursal se realiza el pago de salarios a los dinoempleados que trabajan allí.

El objetivo de esta consulta es obtener información sobre las empresas y el salario promedio que se paga en cada una de ellas.

Nos brinda el nombre de las empresas y el salario promedio que se paga en cada una de ellas. Esto nos permite tener una idea del nivel de remuneración económica que se ofrece en cada empresa del mundo de los dinosaurios.

```sql
SELECT e.nombre_empresa, AVG(p.monto_pago) AS salario_promedio
FROM Empresas e
JOIN Sucursales s ON e.id_empresa = s.empresa_id
JOIN Pagos_Salarios p ON s.id_sucursal = p.sucursal_id
GROUP BY e.id_empresa, e.nombre_empresa;
```

	nombre_empresa	salario_promedio
▶	ecopetrol	2500.000000
	arturo calle	3000.000000
	exito	3000.000000
	movistar	3000.000000
	terpel	3000.000000

movistar

CONSULTA 6

Imaginemos que en el mundo de los dinosaurios existen diferentes departamentos encargados de tareas específicas, como la cría de dinosaurios herbívoros, el cuidado de los dinosaurios carnívoros, la investigación paleontológica, etc. Cada departamento tiene varios dinoempleados que trabajan en él.

El objetivo de esta consulta es obtener información sobre los departamentos y el salario promedio de los dinoempleados que trabajan en cada uno de ellos.

Nos brinda el nombre de los departamentos y el salario promedio de los dinoempleados que trabajan en cada uno de ellos. Esto nos permite tener una idea del nivel salarial en cada departamento del mundo de los dinosaurios y comparar los salarios entre diferentes áreas de trabajo.

```sql
SELECT d.nombre_departamento AS Departamento, AVG(e.salario) AS SalarioPromedio
FROM Empleados e
JOIN Departamentos d ON e.departamento_id = d.id_departamento
GROUP BY d.nombre_departamento;
```

Departamento	SalarioPromedio
Ventas	1003500.000000
recursos	65000.000000
mantenimiento	94000.000000
soporte	6000.000000
ingenerios	99000.000000

CONSULTA 7

IImaginemos que en el mundo de los dinosaurios existen diferentes sucursales encargadas de la crianza y cuidado de dinosaurios. Cada sucursal tiene varios dinoempleados que trabajan en ella y reciben un salario por su labor.

El objetivo de esta consulta es obtener información sobre las sucursales y el salario promedio de los dinoempleados que trabajan en cada una de ellas.

Nos brinda el nombre de las sucursales y el salario promedio de los dinoempleados que trabajan en cada una de ellas. Esto nos permite tener una visión general del nivel salarial en cada sucursal del mundo de los dinosaurios y comparar los salarios entre diferentes ubicaciones.

```sql
SELECT s.nombre_sucursal, AVG(e.salario) AS salario_promedio
FROM Empleados e
JOIN Sucursales s ON e.sucursal_id = s.id_sucursal
GROUP BY s.nombre_sucursal;
```

	nombre_sucursal	salario_promedio
▶	Sucursal A	1003500.000000
	Sucursal B	65000.000000
	Sucursal c	94000.000000
	Sucursal d	6000.000000
	Sucursal e	99000.000000

CONSULTA 8

Imaginemos que en el mundo de los dinosaurios existen diferentes empresas que se dedican a la cría y cuidado de dinosaurios. Cada empresa tiene varias sucursales, y en cada sucursal trabajan dinoempleados que reciben un salario por su labor.

El objetivo de esta consulta es obtener información sobre las empresas y el salario máximo que se paga a los dinoempleados en cada una de ellas.

Nos brinda el nombre de las empresas y el salario máximo que se paga a los dinoempleados en cada una de ellas. Esto nos permite identificar cuál es la empresa que ofrece el mayor salario a sus dinoempleados en el mundo de los dinosaurios.

```sql
SELECT e.nombre_empresa, MAX(em.salario) AS salario_maximo
FROM Empresas e
INNER JOIN Sucursales s ON e.id_empresa = s.empresa_id
INNER JOIN Empleados em ON s.id_sucursal = em.sucursal_id
GROUP BY e.nombre_empresa;
```

nombre_empresa	salario_maximo
ecopetrol	2000000.00
arturo calle	70000.00
exito	100000.00
movistar	6000.00
terpel	99000.00

CONSULTA 9

Supongamos que tenemos una tabla llamada "Empleados" que registra información sobre los dinoempleados de un parque de dinosaurios. Esta tabla contiene columnas como "salario" (que indica el salario de cada empleado) y "departamento_id" (que especifica el departamento al que pertenece cada empleado). Además, tenemos otra tabla llamada "Departamentos" que almacena información sobre los diferentes departamentos del parque, incluyendo la columna "nombre_departamento".

La consulta que mencionas tiene como objetivo encontrar el salario máximo en cada departamento del parque de dinosaurios. Veamos cómo podemos relacionarlo con dinosaurios.

```sql
SELECT d.nombre_departamento, MAX(e.salario) AS salario_maximo
FROM Empleados e
JOIN Departamentos d ON e.departamento_id = d.id_departamento
GROUP BY d.nombre_departamento;
```

	nombre_departamento	salario_maximo
▶	Ventas	2000000.00
	recursos	70000.00
	mantenimiento	100000.00
	soporte	6000.00
	ingenerios	99000.00

CONSULTA 10

Imaginemos que tenemos una tabla llamada "Empleados" que registra información sobre los dinoempleados de un parque temático de dinosaurios. Esta tabla contiene columnas como "salario" (que indica el salario de cada empleado) y "sucursal_id" (que especifica la sucursal a la que pertenece cada empleado). También tenemos otra tabla llamada "Sucursales" que almacena información sobre las diferentes sucursales del parque, incluyendo las columnas "id_sucursal" y "nombre_sucursal".

La consulta que mencionas tiene como objetivo encontrar el salario máximo en cada sucursal del parque de dinosaurios.

```sql
SELECT s.id_sucursal, s.nombre_sucursal, MAX(e.salario) AS salario_maximo
FROM Empleados e
JOIN Sucursales s ON e.sucursal_id = s.id_sucursal
GROUP BY s.id_sucursal, s.nombre_sucursal;
```

id_sucursal	nombre_sucursal	salario_maximo
1	Sucursal A	2000000.00
2	Sucursal B	70000.00
3	Sucursal c	100000.00
4	Sucursal d	6000.00
5	Sucursal e	99000.00

CONSULTA II

La consulta busca obtener el nombre de cada empresa y el salario mínimo entre todos los empleados de sus sucursales.
Imaginemos que las empresas son diferentes especies de dinosaurios, las sucursales son los diferentes hábitats donde habitan y los empleados son los dinosaurios individuales que trabajan en esas sucursales.
La consulta SQL sería como una expedición para descubrir qué especie de dinosaurio tiene el salario mínimo en cada hábitat.
Esta expedición comienza explorando la tabla Empleados, donde encontramos información sobre todos los dinosaurios que trabajan en las sucursales. Luego, nos unimos esa información con la tabla Sucursales, usando el sucursal_id, que es como el número de
identificación del hábitat de cada dinosaurio. Después, conectamos esa información con la tabla Empresas usando el empresa_id, que sería como el número de identificación de la especie de dinosaurio.
Luego, agrupamos los resultados por el nombre de la especie de dinosaurio.
Finalmente, calculamos el salario mínimo para cada grupo de dinosaurios y lo etiquetamos como salario_minimo.
Al final de la expedición, obtendremos un informe que nos muestra el nombre de cada especie de dinosaurio y el salario mínimo entre los dinosaurios que trabajan en los diferentes hábitats.

```sql
SELECT Empresas.nombre_empresa, MIN(Empleados.salario) AS salario_minimo
FROM Empleados
INNER JOIN Sucursales ON Empleados.sucursal_id = Sucursales.id_sucursal
INNER JOIN Empresas ON Sucursales.empresa_id = Empresas.id_empresa
GROUP BY Empresas.nombre_empresa;
```

nombre_empresa	salario_minimo
ecopetrol	7000.00
arturo calle	60000.00
exito	88000.00
movistar	6000.00
terpel	99000.00

CONSULTA 12

LSupongamos que tenemos dos tablas: Departamentos y Empleados. Cada departamento tiene varios empleados.

La consulta busca obtener el nombre de cada departamento y el salario mínimo entre todos los empleados de ese departamento.

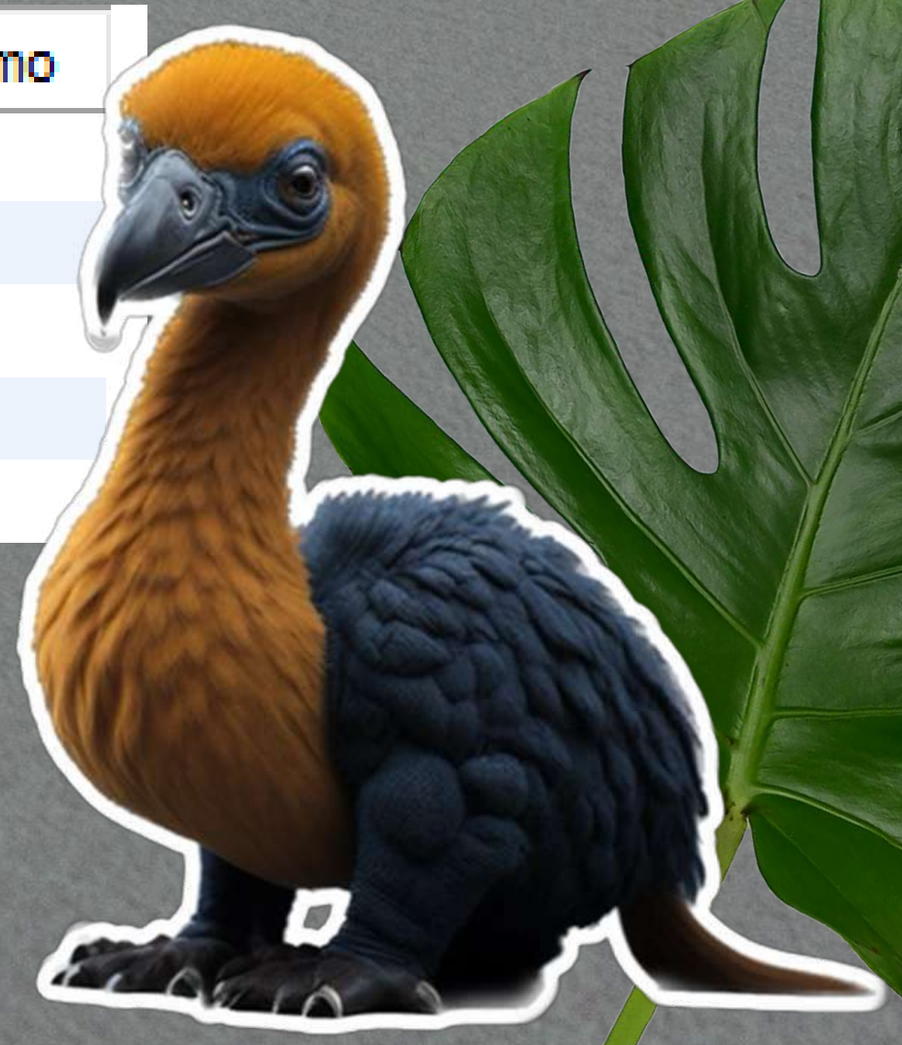
Ahora, imaginemos que los departamentos son diferentes tipos de tareas en una comunidad de dinosaurios y los empleados son los dinosaurios individuales que desempeñan esas tareas.

La consulta sería como una investigación para descubrir qué tipo de tarea tiene el salario mínimo entre los dinosaurios que la realizan.
Esta investigación comienza explorando la tabla Empleados, donde encontramos información sobre todos los dinosaurios y en qué departamento trabajan. Luego, unimos esa información con la tabla Departamentos.
Agrupamos los resultados por el nombre del tipo de tarea. Finalmente, calculamos el salario mínimo para cada grupo de dinosaurios que realizan la misma tarea y lo etiquetamos como salario_minimo.

Al final de la investigación, obtendremos un informe que nos muestra el nombre de cada tipo de tarea o departamento (y el salario mínimo entre los dinosaurios que realizan esa tarea.

```sql
SELECT Departamentos.nombre_departamento, MIN(Empleados.salario) AS salario_minimo
FROM Empleados
INNER JOIN Departamentos ON Empleados.departamento_id = Departamentos.id_departamento
GROUP BY Departamentos.nombre_departamento;
```

	nombre_departamento	salario_minimo
▶	Ventas	7000.00
	recursos	60000.00
	mantenimiento	88000.00
	soporte	6000.00
	ingenerios	99000.00



CONSULTA 13

La consulta busca obtener el nombre de cada sucursal y el salario mínimo entre todos los empleados de esa sucursal.

Ahora, imaginemos que las sucursales son diferentes áreas geográficas habitadas por dinosaurios y los empleados son los dinosaurios individuales que viven en esas áreas.

La consulta sería como una expedición para descubrir qué área geográfica tiene el salario mínimo entre los dinosaurios que viven allí.
Esta expedición comienza explorando la tabla Empleados, donde encontramos información sobre todos los dinosaurios y en qué sucursal viven. Luego, unimos esa información con la tabla Sucursales.
Agrupamos los resultados por el nombre de la sucursal. Finalmente, calculamos el salario mínimo para cada grupo de dinosaurios que viven en la misma área geográfica y lo etiquetamos como salario_minimo.

Al final de la expedición, obtendremos un informe que nos muestra el nombre de cada área geográfica o sucursal y el salario mínimo entre los dinosaurios que viven en esa área geográfica.

```sql
SELECT Sucursales.nombre_sucursal, MIN(Empleados.salario) AS salario_minimo
FROM Empleados
INNER JOIN Sucursales ON Empleados.sucursal_id = Sucursales.id_sucursal
GROUP BY Sucursales.nombre_sucursal;
```

nombre_sucursal	salario_minimo
Sucursal A	00
Sucursal B	60000.00
Sucursal c	88000.00
Sucursal d	6000.00
Sucursal e	99000.00

Sucursal A

CONSULTA 14

Esta consulta busca obtener el nombre de cada hábitat y el número de dinosaurios cuyo tamaño es superior al promedio de su especie en ese hábitat.

Se realiza una unión entre las tablas Dinosaurios y Hábitats. Luego, se aplica un filtro para seleccionar solo los dinosaurios cuyo tamaño es mayor al promedio de su especie en ese hábitat.

Finalmente, se realiza una agrupación por el nombre del hábitat y se cuenta el número de dinosaurios en cada hábitat.

El resultado final es un informe que muestra el nombre de cada hábitat y el número de dinosaurios que tienen un tamaño superior al promedio de su especie en ese hábitat.

```sql
SELECT d.nombre_departamento AS departamento, COUNT(*) AS num_empleados
FROM Empleados e
INNER JOIN Departamentos d ON e.departamento_id = d.id_departamento
WHERE e.salario > (
  SELECT AVG(e2.salario)
  FROM Empleados e2
  WHERE e2.departamento_id = e.departamento_id
  AND e2.id_empleado != e.id_empleado
)
GROUP BY d.nombre_departamento;
```

	nombre_departamento	cantidad_empleados
▶	Ventas	1
	recursos	1
	mantenimiento	1

CONSULTA 15

Esta consulta busca obtener el número de empleados y el identificador de la sucursal para aquellos casos en los que el monto de pago en la tabla es mayor al salario promedio en cada sucursal.

Ahora, imaginemos que los pagos de salarios son recompensas que reciben los dinosaurios empleados y las sucursales son diferentes áreas de trabajo donde los dinosaurios están asignados.

Esta investigación comienza explorando la tabla Pagos_Salarios. Luego, realizamos una unión entre esa información y una subconsulta. Esta subconsulta calcula el salario promedio para cada área de trabajo o sucursal. En la subconsulta, también se involucran las tablas Empleados y Departamentos para obtener información adicional.

Luego, se compara el monto de pago en cada fila de Pagos_Salarios con el salario promedio correspondiente a la sucursal en la que trabaja el dinosaurio empleado.

Después de eso, se agrupan los resultados por el identificador de la sucursal en Pagos_Salarios y se cuenta el número de empleados para cada sucursal.

Al final de la investigación, obtendremos un informe que muestra el número de empleados y el identificador de la sucursal para aquellos casos en los que el monto de pago sea mayor al salario promedio en cada área de trabajo o sucursal.

```sql
SELECT COUNT(*) AS num_empleados, Pagos_Salarios.sucursal_id
FROM Pagos_Salarios
INNER JOIN (
    SELECT sucursal_id, AVG(salario) AS salario_promedio
    FROM Empleados
    INNER JOIN Departamentos ON Empleados.departamento_id = Departamentos.id_departamento
    GROUP BY sucursal_id
) AS subquery ON Pagos_Salarios.sucursal_id = subquery.sucursal_id
WHERE Pagos_Salarios.monto_pago > subquery.salario_promedio
GROUP BY Pagos_Salarios.sucursal_id;
```

	nombre_sucursal	numero_empleados
▶	Sucursal A	1
	Sucursal B	1
	Sucursal c	1

CONSULTA 16

La consulta busca obtener el número de empleados y el identificador de la empresa para aquellos casos en los que el monto de pago sea mayor al salario promedio en cada empresa.

Ahora, imaginemos que los pagos de salarios son recompensas que reciben los dinosaurios empleados, las sucursales son áreas de trabajo donde los dinosaurios están asignados, y las empresas son agrupaciones más grandes que contienen varias sucursales.

La consulta es una investigación para identificar en qué empresas los dinosaurios empleados reciben un pago mayor al promedio en su empresa.
Esta investigación comienza explorando la tabla Pagos_Salarios. Luego, realizamos una serie de uniones con las tablas Empleados, Sucursales y una subconsulta.

En la subconsulta, se calcula el salario promedio para cada empresa, los pagos de salarios y las sucursales relacionadas.

Se compara el monto de pago con el salario promedio correspondiente a la empresa en la que trabaja el dinosaurio empleado.

Se agrupan los resultados y se cuenta el número de empleados para cada empresa.

Al final de la investigación, obtendremos un informe que muestra el número de empleados y el identificador de la empresa para aquellos casos en los que el monto de pago sea mayor al salario promedio en cada empresa.

```sql
SELECT COUNT(*) AS num_empleados, Sucursales.empresa_id
FROM Pagos_Salarios
INNER JOIN Empleados ON Pagos_Salarios.empleado_id = Empleados.id_empleado
INNER JOIN Sucursales ON Empleados.sucursal_id = Sucursales.id_sucursal
INNER JOIN (
    SELECT empresa_id, AVG(salario) AS salario_promedio
    FROM Empleados
    INNER JOIN Pagos_Salarios ON Empleados.id_empleado = Pagos_Salarios.empleado_id
    INNER JOIN Sucursales ON Empleados.sucursal_id = Sucursales.id_sucursal
    GROUP BY empresa_id
) AS subquery ON Sucursales.empresa_id = subquery.empresa_id
WHERE Pagos_Salarios.monto_pago > subquery.salario_promedio
GROUP BY Sucursales.empresa_id;
```

	nombre_empresa	numero_empleados
▶	ecopetrol	1
	arturo calle	1
	exito	1

CONSULTA 17

La consulta busca obtener el nombre de cada departamento de dinosaurios y el total de recompensas pagadas en el último mes.
Imaginemos que los pagos de salarios son recompensas que reciben los dinosaurios empleados por su arduo trabajo. Cada dinosaurio empleado está asignado a un departamento en el que trabajan juntos.

La consulta sería como una expedición para descubrir cuántas recompensas se han pagado en el último mes para cada departamento de dinosaurios.

Al final de esta emocionante expedición, obtenemos un informe que muestra el nombre de cada departamento de dinosaurios y el total de recompensas pagadas en el último mes para ese departamento.

```sql
SELECT d.nombre_departamento, SUM(ps.monto_pago) AS total_salarios_pagados
FROM Pagos_Salarios ps
JOIN Empleados e ON e.id_empleado = ps.empleado_id
JOIN Departamentos d ON d.id_departamento = ps.departamento_id
WHERE ps.fecha_pago >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
GROUP BY d.nombre_departamento;
```

	nombre_departamento	total_salarios_pagados
▶	Ventas	2500.00
	recursos	3000.00
	mantenimiento	3000.00
	soporte	00.00
	ingenerios	3000.00

soporte

CONSULTA 18

La consulta busca obtener el identificador de la manada de dinosaurios , el nombre de la manada y el total de presas capturadas en el último mes. Se realiza una manada entre las tablas Pagos_Salarios, Sucursales y Empresas.

La cláusula INNER JOIN es como una manada de dinosaurios trabajando juntos. Une las filas de las tablas según las condiciones establecidas.
La condición WHERE es como una caza en el tiempo. Filtra las presas según la fecha de pago. Se seleccionan solo los pagos realizados en el último mes utilizando el rango de tiempo desde hace un mes hasta el presente.

Luego, se realiza una agrupación por el identificador de la manada y el nombre de la manada.
El resultado final será un informe que muestra el identificador de la manada, el nombre de la manada y el total de presas capturadas en el último mes para cada manada de dinosaurios.

```sql
SELECT s.empresa_id, e.nombre_empresa, SUM(p.monto_pago) as total_salarios
FROM Pagos_Salarios p
INNER JOIN Sucursales s ON p.sucursal_id = s.id_sucursal
INNER JOIN Empresas e ON s.empresa_id = e.id_empresa
WHERE p.fecha_pago BETWEEN DATE_SUB(NOW(), INTERVAL 1 MONTH) AND NOW()
GROUP BY s.empresa_id, e.nombre_empresa
```

empresa_id	nombre_empresa	total_salarios
1	ecopetrol	2500.00
2	arturo calle	3000.00
3	exito	3000.00
4	movistar	3000.00
5	terpel	3000.00

CONSULTA 19

La consulta busca obtener el identificador del dinosaurio empleado y el total de presas capturadas en el último mes. Se seleccionan los datos de la tabla Pagos_Salarios.

Imaginemos que los pagos de salarios son recompensas que reciben los dinosaurios empleados por cazar y capturar presas. Cada dinosaurio empleado tiene un identificador único que los distingue.

La consulta sería como una expedición para descubrir cuántas presas ha capturado cada dinosaurio empleado en el último mes.
Al final de esta emocionante expedición, obtenemos un informe que muestra el identificador de cada dinosaurio empleado y el total de presas capturadas en el último mes para ese dinosaurio empleado.

```
SELECT empleado_id, SUM(monto_pago) as total_salarios
FROM Pagos_Salarios
WHERE fecha_pago >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
GROUP BY empleado_id;
```

	nombre_departamento	total_salarios_pagados
▶	Ventas	2500.00
	recursos	3000.00
	mantenimiento	3000.00
	soporte	00.00
	ingenerios	3000.00

soporte

CONSULTA 20

Imagina que tenemos un grupo de empresas que representan diferentes especies de dinosaurios. Cada empresa tiene varias sucursales, que podríamos comparar con las diferentes regiones habitadas por los dinosaurios de esa especie. Dentro de cada sucursal, hay empleados que representan los dinosaurios individuales.

Ahora, supongamos que estas empresas tienen un sistema de bonificaciones que premia a los dinosaurios que han hecho un buen trabajo. Queremos saber cuántos dinosaurios de cada especie han recibido bonificaciones en el último mes.

```sql
SELECT Empresas.nombre_empresa, COUNT(DISTINCT Bonificaciones.empleado_id) AS numero_empleados
FROM Empresas
INNER JOIN Sucursales ON Empresas.id_empresa = Sucursales.empresa_id
INNER JOIN Empleados ON Sucursales.id_sucursal = Empleados.sucursal_id
INNER JOIN Bonificaciones ON Empleados.id_empleado = Bonificaciones.empleado_id
WHERE Bonificaciones.fecha_bonificacion >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
GROUP BY Empresas.nombre_empresa;
```

nombre_empresa	numero_empleados
arturo calle	1
ecopetrol	1
exito	1

CONSULTA 21

Imaginemos que en nuestra organización hay varios departamentos, y cada departamento se especializa en un tipo de alimento para los dinosaurios, como "Herbívoros", "Carnívoros" y "Omnívoros". Los empleados representan a los dinosaurios que trabajan en cada departamento.

En este caso, estamos uniendo las tablas "Empleados" y "Bonificaciones" mediante las claves correspondientes para relacionar los empleados con las bonificaciones que han recibido. Utilizamos la cláusula JOIN para combinar la información de ambas tablas.

```sql
SELECT E.departamento_id, COUNT(B.empleado_id) AS numero_empleados
FROM Empleados AS E
JOIN Bonificaciones AS B ON E.id_empleado = B.empleado_id
WHERE B.fecha_bonificacion >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
GROUP BY E.departamento_id;
```

departamento_id	numero_empleados
1	1
2	1
3	1

CONSULTA 22

el objetivo es obtener el número de empleados que han recibido bonificaciones en el último mes, agrupados por sucursal. Vamos a representar las sucursales como diferentes hábitats para los dinosaurios.

Imaginemos que tenemos una organización con varias sucursales, y cada sucursal es como un hábitat específico para los dinosaurios. Los empleados representan a los dinosaurios individuales que trabajan en cada sucursal.

En esta consulta, estamos uniendo las tablas "Empleados" y "Bonificaciones" utilizando las claves correspondientes para relacionar los empleados con las bonificaciones que han recibido. Utilizamos la cláusula JOIN para combinar la información de ambas tablas.

Luego, utilizamos la cláusula WHERE para filtrar las bonificaciones en función de la fecha. Estamos buscando las bonificaciones que hayan ocurrido en el último mes utilizando la función DATE_SUB(CURDATE(), INTERVAL 1 MONTH), que resta un mes a la fecha actual.

Después, utilizamos GROUP BY para agrupar los resultados por la sucursal a la que pertenece cada empleado. Esto nos permitirá obtener el número de empleados que han recibido bonificaciones en el último mes para cada sucursal.

```sql
SELECT E.sucursal_id, COUNT(B.empleado_id) AS numero_empleados
FROM Empleados AS E
JOIN Bonificaciones AS B ON E.id_empleado = B.empleado_id
WHERE B.fecha_bonificacion >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
GROUP BY E.sucursal_id;
```

sucursal_id	numero_empleados
1	1
2	1
3	1

CONSULTA 23

Imaginemos que tenemos varias empresas, y cada empresa es como un grupo de dinosaurios de una especie particular. Las sucursales serían diferentes regiones donde se encuentran estos grupos de dinosaurios, y los empleados representarían a los dinosaurios individuales dentro de cada sucursal.

En esta consulta, estamos uniendo las tablas "Empresas", "Sucursales", "Empleados" y "Bonificaciones" utilizando las claves correspondientes para relacionar la información. Utilizamos la cláusula JOIN para combinar los datos de estas tablas.

Luego, utilizamos la cláusula WHERE para filtrar las bonificaciones en función de la fecha. Estamos buscando las bonificaciones que hayan ocurrido en el último mes utilizando la función DATE_SUB(CURDATE(), INTERVAL 1 MONTH), que resta un mes a la fecha actual.

Después, utilizamos GROUP BY para agrupar los resultados por el nombre de la empresa. Esto nos permitirá obtener el total de bonificaciones otorgadas en el último mes para cada empresa.

```sql
SELECT Empresas.nombre_empresa, SUM(Bonificaciones.monto_bonificacion) AS total_bonificaciones
FROM Empresas
INNER JOIN Sucursales ON Empresas.id_empresa = Sucursales.empresa_id
INNER JOIN Empleados ON Sucursales.id_sucursal = Empleados.sucursal_id
INNER JOIN Bonificaciones ON Empleados.id_empleado = Bonificaciones.empleado_id
WHERE Bonificaciones.fecha_bonificacion >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
GROUP BY Empresas.nombre_empresa;
```

nombre_empresa	total_bonificaciones
ecopetrol	200.00
arturo calle	350.00
exito	450.00

CONSULTA 24

Imaginemos que en nuestra organización hay varios departamentos, y cada departamento se especializa en un tipo de alimento para los dinosaurios, como "Herbívoros", "Carnívoros" y "Omnívoros". Los empleados representan a los dinosaurios que trabajan en cada departamento.

En esta consulta, estamos uniendo las tablas "Bonificaciones" y "Empleados" utilizando las claves correspondientes para relacionar los datos. Utilizamos la cláusula JOIN para combinar la información de estas tablas.

Luego, utilizamos la cláusula WHERE para filtrar las bonificaciones en función de la fecha. Estamos buscando las bonificaciones que hayan ocurrido en el último mes utilizando la función DATE_SUB(CURDATE(), INTERVAL 1 MONTH), que resta un mes a la fecha actual.

Después, utilizamos GROUP BY para agrupar los resultados por el departamento al que pertenece cada empleado. Esto nos permitirá obtener el total de bonificaciones otorgadas en el último mes para cada departamento.

```sql
SELECT E.departamento_id, SUM(B.monto_bonificacion) AS total_bonificaciones
FROM Bonificaciones AS B
JOIN Empleados AS E ON B.empleado_id = E.id_empleado
WHERE B.fecha_bonificacion >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
GROUP BY E.departamento_id;
```

departamento_id	total_bonificaciones
1	200.00
2	350.00
3	450.00

CONSULTA 25

Imaginemos que tenemos una organización con varias sucursales, y cada sucursal es como un hábitat específico para los dinosaurios. Los empleados representan a los dinosaurios individuales que trabajan en cada sucursal.

En esta consulta, estamos uniendo las tablas "Bonificaciones" y "Empleados" utilizando las claves correspondientes para relacionar los datos. Utilizamos la cláusula JOIN para combinar la información de estas tablas.

Luego, utilizamos la cláusula WHERE para filtrar las bonificaciones en función de la fecha. Estamos buscando las bonificaciones que hayan ocurrido en el último mes utilizando la función DATE_SUB(CURDATE(), INTERVAL 1 MONTH), que resta un mes a la fecha actual.

Después, utilizamos GROUP BY para agrupar los resultados por la sucursal a la que pertenece cada empleado. Esto nos permitirá obtener el total de bonificaciones otorgadas en el último mes para cada sucursal.

```
1
2 •   SELECT E.sucursal_id, SUM(B.monto_bonificacion) AS total_bonificaciones
3     FROM Bonificaciones AS B
4     JOIN Empleados AS E ON B.empleado_id = E.id_empleado
5     WHERE B.fecha_bonificacion >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
6     GROUP BY E.sucursal_id;
7
```

sucursal_id	total_bonificaciones
1	200.00
2	350.00
3	450.00

CONSULTA 26

Esta consulta tiene como objetivo contar el número de empleados que han tenido aumentos salariales de más del 10% en los últimos 6 meses. No utilizaremos la metáfora de dinosaurios en esta explicación, ya que la consulta se centra en la cantidad de empleados que cumplen ciertos criterios.

En la consulta, se realiza una unión (JOIN) entre las tablas "Aumentos_Salariales" y "Empleados" utilizando la clave correspondiente para relacionar los datos. La cláusula ON especifica que la columna "empleado_id" de la tabla "Aumentos_Salariales" debe coincidir con la columna "id_empleado" de la tabla "Empleados".

A continuación, se utiliza la cláusula WHERE para filtrar los registros. Se buscan los aumentos salariales donde el porcentaje de aumento (porcentaje_aumento) sea mayor que 10 y la fecha de aumento (fecha_aumento) sea mayor o igual a 6 meses atrás a partir de la fecha actual. Esto se logra utilizando la función DATE_SUB(CURDATE(), INTERVAL 6 MONTH).

Por último, se utiliza COUNT(*) para contar el número de filas resultantes de la consulta, lo que nos dará el número de empleados que han tenido aumentos salariales de más del 10% en los últimos 6 meses.

```sql
SELECT COUNT(*) AS numero_empleados
FROM Aumentos_Salariales AS A
JOIN Empleados AS E ON A.empleado_id = E.id_empleado
WHERE A.porcentaje_aumento > 10 AND A.fecha_aumento >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH);
```

numero_empleados
2

CONSULTA 26.1

Esta consulta tiene como objetivo contar el número de empleados que han tenido aumentos salariales de menos del 5% en los últimos 6 meses. No utilizaremos la metáfora de dinosaurios en esta explicación, ya que la consulta se centra en la cantidad de empleados que cumplen ciertos criterios.

En la consulta, se realiza una unión (JOIN) entre las tablas "Aumentos_Salariales" y "Empleados" utilizando la clave correspondiente para relacionar los datos. La cláusula ON especifica que la columna "empleado_id" de la tabla "Aumentos_Salariales" debe coincidir con la columna "id_empleado" de la tabla "Empleados".

A continuación, se utiliza la cláusula WHERE para filtrar los registros. Se buscan los aumentos salariales donde el porcentaje de aumento (porcentaje_aumento) sea menor que 5 y la fecha de aumento (fecha_aumento) sea mayor o igual a 6 meses atrás a partir de la fecha actual. Esto se logra utilizando la función DATE_SUB(CURDATE(), INTERVAL 6 MONTH).

Por último, se utiliza COUNT(*) para contar el número de filas resultantes de la consulta, lo que nos dará el número de empleados que han tenido aumentos salariales de menos del 5% en los últimos 6 meses.

```sql
SELECT COUNT(*) AS numero_empleados
FROM Aumentos_Salariales AS A
JOIN Empleados AS E ON A.empleado_id = E.id_empleado
WHERE A.porcentaje_aumento < 5 AND A.fecha_aumento >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH);
```

	numero_empleados
	1

CONSULTA 27

Imaginemos que los empleados de la empresa son como diferentes especies de dinosaurios, cada uno con su propia capacidad para trabajar. Algunos empleados son como dinosaurios feroces que muestran un gran rendimiento en su trabajo.

En esta consulta, se utiliza la tabla "Horas_Trabajadas" que representa las horas trabajadas por cada empleado. Utilizamos la cláusula WHERE para filtrar los registros. Estamos buscando las horas trabajadas donde el número de horas (horas) sea mayor a 160, lo cual indica que los empleados han trabajado de manera "feroz" durante un periodo de tiempo específico.

Además, utilizamos Horas_Trabajadas.fecha_trabajo >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH) para asegurarnos de que solo se consideren las horas trabajadas en el último mes.

Finalmente, utilizamos COUNT(DISTINCT Horas_Trabajadas.empleado_id) para contar el número de empleados únicos que han tenido un desempeño laboral "feroz" en el último mes.

```sql
SELECT COUNT(DISTINCT Horas_Trabajadas.empleado_id) AS numero_empleados
FROM Horas_Trabajadas
WHERE Horas_Trabajadas.horas > 160
  AND Horas_Trabajadas.fecha_trabajo >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH);
```

numero_empleados
1

CONSULTA 28

Imaginemos que los empleados de la empresa son como diferentes especies de dinosaurios, cada uno con su propia capacidad para trabajar. Algunos empleados son como dinosaurios tranquilos que realizan un trabajo más relajado y no acumulan muchas horas de trabajo.

En esta consulta, se utiliza la tabla "Horas_Trabajadas" que representa las horas trabajadas por cada empleado. Utilizamos la cláusula WHERE para filtrar los registros. Estamos buscando las horas trabajadas donde el número de horas (horas) sea menor a 80, lo cual indica que los empleados han tenido un desempeño laboral más "tranquilo" durante un periodo de tiempo específico.

Además, utilizamos Horas_Trabajadas.fecha_trabajo >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH) para asegurarnos de que solo se consideren las horas trabajadas en el último mes.

Finalmente, utilizamos COUNT(DISTINCT Horas_Trabajadas.empleado_id) para contar el número de empleados únicos que han tenido un desempeño laboral "tranquilo" (menos de 80 horas trabajadas) en el último mes.

```sql
SELECT COUNT(DISTINCT Horas_Trabajadas.empleado_id) AS numero_empleados
FROM Horas_Trabajadas
WHERE Horas_Trabajadas.horas < 80
  AND Horas_Trabajadas.fecha_trabajo >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH);
```

numero_empleados
1

CONSULTA 29

En la tabla "Empleados", cada registro representa a un empleado, y cada empleado es como un dinosaurio individual. Tenemos una columna llamada "género" que representa el género de cada empleado, que podemos equiparar a los diferentes géneros de dinosaurios en nuestro escenario.

Al usar la cláusula GROUP BY, estamos agrupando los resultados según el campo "género". Esto es similar a agrupar a los dinosaurios por su género, es decir, reunir a todos los dinosaurios del mismo género juntos.

Luego, utilizamos la función de agregación AVG(salario) para calcular el salario promedio dentro de cada grupo. Esto nos permitirá determinar el salario promedio para cada género de dinosaurio en la empresa.

```sql
SELECT género, AVG(salario) AS salario_promedio
FROM Empleados
GROUP BY género;
```

genero	salario_promedio
Masculino	544250.000000
Femenino	63250.000000

CONSULTA 30

Vamos a imaginar que los empleados representan diferentes especies de dinosaurios y que cada grupo de edad de dinosaurio tiene sus propios niveles salariales.

En la tabla "Empleados", cada registro representa a un empleado, y cada empleado es como un dinosaurio individual. Tenemos una columna llamada "edad" que representa la edad de cada empleado, que podemos equiparar a los diferentes grupos de edad de dinosaurios en nuestro escenario.

Al usar la cláusula GROUP BY, estamos agrupando los resultados según el campo "edad". Esto es similar a agrupar a los dinosaurios por su grupo de edad, es decir, reunir a todos los dinosaurios del mismo grupo de edad juntos.

Luego, utilizamos la función de agregación AVG(salario) para calcular el salario promedio dentro de cada grupo de edad. Esto nos permitirá determinar el salario promedio para cada grupo de edad de dinosaurios en la empresa.

```sql
SELECT edad, AVG(salario) AS salario_promedio
FROM Empleados
GROUP BY edad;
```

edad	salario_promedio
30	7000.000000
22	60000.000000
25	93500.000000
18	6000.000000
19	70000.000000
21	100000.000000
20	2000000.000000

CONSULTA 31

Imaginemos que los empleados representan diferentes especies de dinosaurios y que cada nivel educativo de dinosaurio tiene sus propios niveles salariales.

En la tabla "Empleados", cada registro representa a un empleado, y cada empleado es como un dinosaurio individual. Tenemos una columna llamada "nivel_educativo" que representa el nivel de educación de cada empleado, que podemos equiparar a los diferentes niveles educativos de los dinosaurios en nuestro escenario.

Al utilizar la cláusula GROUP BY, estamos agrupando los resultados según el campo "nivel_educativo". Esto es similar a agrupar a los dinosaurios según su nivel educativo, es decir, reunir a todos los dinosaurios con el mismo nivel educativo juntos.

Luego, utilizamos la función de agregación AVG(salario) para calcular el salario promedio dentro de cada grupo de nivel educativo. Esto nos permitirá determinar el salario promedio para cada nivel educativo de dinosaurios en la empresa.

```
SELECT nivel_educativo, AVG(salario) AS salario_promedio
FROM Empleados
GROUP BY nivel_educativo;
```

nivel_educativo	salario_promedio
Licenciatura	7000.000000
Maestría	60000.000000
tecnologo	1044000.000000
bachillerato	6000.000000
profesional	99000.000000
estudiante	70000.000000
tecnologo	100000.000000

CONSULTA 32

El objetivo de esta consulta es contar el número de empleados con más de 5 años de experiencia laboral en cada sucursal de la empresa. Vamos a imaginar que cada sucursal es como un hábitat diferente donde los dinosaurios trabajan y viven.

En la tabla "Empleados", cada registro representa a un empleado, que sería como un dinosaurio individual. Tenemos una columna llamada "experiencia_laboral" que representa la experiencia laboral de cada empleado, que podemos equiparar a la edad o los conocimientos acumulados por los dinosaurios en nuestro escenario.

En la tabla "Sucursales", cada registro representa una sucursal de la empresa, que podemos considerar como diferentes hábitats donde trabajan los dinosaurios.

Al utilizar la cláusula JOIN, unimos la tabla "Empleados" con la tabla "Sucursales" utilizando el campo "sucursal_id", de manera similar a cómo los dinosaurios pueden pertenecer y trabajar en diferentes sucursales según su ubicación.

Luego, utilizamos la cláusula WHERE para filtrar los registros y seleccionar solo aquellos empleados con más de 5 años de experiencia laboral, es decir, aquellos dinosaurios que han acumulado una experiencia laboral significativa.

Utilizando la cláusula GROUP BY S.nombre sucursal, agrupamos los resultados por el nombre de la sucursal, lo cual nos permite contar el número de empleados con más de 5 años de experiencia laboral en cada sucursal.

```sql
SELECT D.nombre_departamento, COUNT(*) AS numero_empleados
FROM Empleados AS E
JOIN Departamentos AS D ON E.departamento_id = D.id_departamento
WHERE E.experiencia_laboral > 10
GROUP BY D.nombre_departamento;
```


	nombre_departamento	numero_empleados
▶	Ventas	1
	recursos	1

CONSULTA 33

imaginar que cada departamento es como un hábitat diferente donde los dinosaurios trabajan y que algunos de ellos son recién nacidos sin experiencia laboral.

En la tabla "Empleados", cada registro representa a un empleado, que sería como un dinosaurio individual. La columna "experiencia_laboral" representa la experiencia laboral de cada empleado, que en este caso, si es menor a 1, significa que no tienen experiencia laboral previa.

En la tabla "Departamentos", cada registro representa un departamento de la empresa, que se puede considerar como diferentes hábitats donde trabajan los dinosaurios.

Al utilizar la cláusula JOIN, unimos la tabla "Empleados" con la tabla "Departamentos" utilizando el campo "departamento_id", de manera similar a cómo los dinosaurios pueden pertenecer y trabajar en diferentes departamentos según su especialización.

Luego, utilizando la cláusula WHERE, filtramos los registros y seleccionamos aquellos empleados cuya experiencia laboral es menor a 1, es decir, aquellos dinosaurios que son recién nacidos sin experiencia laboral previa.

Utilizando la cláusula GROUP BY D.nombre_departamento, agrupamos los resultados por el nombre del departamento, lo cual nos permite contar el número de empleados sin experiencia laboral en cada departamento.

```sql
SELECT D.nombre_departamento, COUNT(*) AS numero_empleados
FROM Empleados AS E
JOIN Departamentos AS D ON E.departamento_id = D.id_departamento
WHERE E.experiencia_laboral < 1
GROUP BY D.nombre_departamento;
```

nombre_departamento	numero_empleados
mantenimiento	1
ingenerios	1

CONSULTA 34

Vamos a imaginar que cada departamento es como un hábitat diferente donde los dinosaurios trabajan y que la experiencia laboral de los dinosaurios se mide en años.

En la tabla "Empleados", cada registro representa a un empleado, que sería como un dinosaurio individual. La columna "experiencia_laboral" representa la experiencia laboral de cada empleado, que en este caso, si está entre 1 y 5 años, significa que tienen cierta experiencia laboral acumulada.

En la tabla "Departamentos", cada registro representa un departamento de la empresa, que se puede considerar como diferentes hábitats donde trabajan los dinosaurios.

Al utilizar la cláusula JOIN, unimos la tabla "Empleados" con la tabla "Departamentos" utilizando el campo "departamento_id", de manera similar a cómo los dinosaurios pueden pertenecer y trabajar en diferentes departamentos según su especialización.

Luego, utilizando la cláusula WHERE, filtramos los registros y seleccionamos aquellos empleados cuya experiencia laboral está entre 1 y 5 años, es decir, aquellos dinosaurios que tienen cierta experiencia laboral acumulada pero no son ni recién nacidos sin experiencia laboral, ni dinosaurios con una experiencia laboral muy extensa.

Utilizando la cláusula GROUP BY D.nombre_departamento, agrupamos los resultados por el nombre del departamento, lo cual nos permite contar el número de empleados con experiencia laboral de entre 1 y 5 años en cada departamento.

```sql
SELECT D.nombre_departamento, COUNT(*) AS numero_empleados
FROM Empleados AS E
JOIN Departamentos AS D ON E.departamento_id = D.id_departamento
WHERE E.experiencia_laboral BETWEEN 1 AND 5
GROUP BY D.nombre_departamento;
```

nombre_departamento	numero_empleados
recursos	1
Ventas	1

CONSULTA 35

Vamos a imaginar que cada departamento es como un hábitat diferente donde los dinosaurios trabajan y que la experiencia laboral de los dinosaurios se mide en años.

En la tabla "Empleados", cada registro representa a un empleado, que sería como un dinosaurio individual. La columna "experiencia_laboral" representa la experiencia laboral de cada empleado, que en este caso, si está entre 5 y 10 años, significa que tienen una experiencia laboral considerable acumulada.

En la tabla "Departamentos", cada registro representa un departamento de la empresa, que se puede considerar como diferentes hábitats donde trabajan los dinosaurios.

Al utilizar la cláusula JOIN, unimos la tabla "Empleados" con la tabla "Departamentos" utilizando el campo "departamento_id", de manera similar a cómo los dinosaurios pueden pertenecer y trabajar en diferentes departamentos según su especialización.

Luego, utilizando la cláusula WHERE, filtramos los registros y seleccionamos aquellos empleados cuya experiencia laboral está entre 5 y 10 años, es decir, aquellos dinosaurios que tienen una experiencia laboral significativa pero no son dinosaurios con una experiencia laboral muy extensa.

Utilizando la cláusula GROUP BY D.nombre_departamento, agrupamos los resultados por el nombre del departamento, lo cual nos permite contar el número de empleados con experiencia laboral de entre 5 y 10 años en cada departamento.

```sql
SELECT D.nombre_departamento, COUNT(*) AS numero_empleados
FROM Empleados AS E
JOIN Departamentos AS D ON E.departamento_id = D.id_departamento
WHERE E.experiencia_laboral BETWEEN 5 AND 10
GROUP BY D.nombre_departamento;
```

nombre_departamento	numero_empleados
soporte	1
mantenimiento	1
Ventas	1

CONSULTA 36

Vamos a imaginar que cada departamento es como un hábitat diferente donde los dinosaurios trabajan y que los dinosaurios tienen diferentes géneros, al igual que los empleados tienen diferentes géneros.

En la tabla "Empleados", cada registro representa a un empleado, que sería como un dinosaurio individual. La columna "salario" representa el salario de cada empleado, que en este caso, sería como la cantidad de recursos que los dinosaurios obtienen en su hábitat.

En la tabla "Departamentos", cada registro representa un departamento de la empresa, que se puede considerar como diferentes hábitats donde trabajan los dinosaurios.

Al utilizar la cláusula JOIN, unimos la tabla "Empleados" con la tabla "Departamentos" utilizando el campo "departamento_id", de manera similar a cómo los dinosaurios pueden pertenecer y trabajar en diferentes departamentos según su especialización.

Utilizando la cláusula GROUP BY D.nombre_departamento, E.género, agrupamos los resultados por el nombre del departamento y el género de los empleados, lo cual nos permite calcular el salario promedio para cada combinación de departamento y género.

```sql
SELECT D.nombre_departamento, E.género, AVG(E.salario) AS salario_promedio
FROM Empleados AS E
JOIN Departamentos AS D ON E.departamento_id = D.id_departamento
GROUP BY D.nombre_departamento, E.género;
```

nombre_departamento	genero	salario_promedio
Ventas	Masculino	1003500.000000
recursos	Femenino	60000.000000
recursos	masculino	70000.000000
mantenimiento	Femenino	88000.000000
mantenimiento	masculino	100000.000000
soporte	Femenino	6000.000000
ingenerios	Femenino	99000.000000

CONSULTA 37

Vamos a imaginar que cada sucursal es como un hábitat diferente donde los dinosaurios trabajan y que los dinosaurios tienen diferentes géneros, al igual que los empleados tienen diferentes géneros.

En la tabla "Empleados", cada registro representa a un empleado, que sería como un dinosaurio individual. La columna "salario" representa el salario de cada empleado, que en este caso, sería como la cantidad de recursos que los dinosaurios obtienen en su hábitat.

En la tabla "Sucursales", cada registro representa una sucursal de la empresa, que se puede considerar como diferentes hábitats donde trabajan los dinosaurios.

Al utilizar la cláusula JOIN, unimos la tabla "Empleados" con la tabla "Sucursales" utilizando el campo "sucursal_id", de manera similar a cómo los dinosaurios pueden pertenecer y trabajar en diferentes sucursales según su ubicación.

Utilizando la cláusula GROUP BY S.nombre_sucursal, E.género, agrupamos los resultados por el nombre de la sucursal y el género de los empleados, lo cual nos permite calcular el salario promedio para cada combinación de sucursal y género.

```sql
SELECT S.nombre_sucursal, E.género, AVG(E.salario) AS salario_promedio
FROM Empleados AS E
JOIN Sucursales AS S ON E.sucursal_id = S.id_sucursal
GROUP BY S.nombre_sucursal, E.género;
```

nombre_sucursal	genero	salario_promedio
Sucursal A	Masculino	1003500.000000
Sucursal B	Femenino	60000.000000
Sucursal B	masculino	70000.000000
Sucursal c	Femenino	88000.000000
Sucursal c	masculino	100000.000000
Sucursal d	Femenino	6000.000000
Sucursal e	Femenino	99000.000000

CONSULTA 38

Vamos a imaginar que cada empresa es como un ecosistema completo donde los dinosaurios trabajan, y dentro de cada empresa hay diferentes sucursales que representan hábitats más específicos. Además, los dinosaurios tienen diferentes géneros, al igual que los empleados tienen diferentes géneros.

En la tabla "Empleados", cada registro representa a un empleado, que sería como un dinosaurio individual. La columna "salario" representa el salario de cada empleado, que en este caso, sería como la cantidad de recursos que los dinosaurios obtienen en su ecosistema.

En la tabla "Sucursales", cada registro representa una sucursal de la empresa, que se puede considerar como diferentes hábitats dentro del ecosistema de la empresa.

En la tabla "Empresas", cada registro representa una empresa completa, que sería como el ecosistema general donde trabajan los dinosaurios.

Al utilizar la cláusula JOIN, unimos la tabla "Empleados" con la tabla "Sucursales" utilizando el campo "sucursal_id", y luego unimos la tabla "Sucursales" con la tabla "Empresas" utilizando el campo "empresa_id". Esto nos permite conectar a los empleados con su respectiva empresa y sucursal en el ecosistema laboral de los dinosaurios.

Utilizando la cláusula GROUP BY EM.nombre_empresa, E.género, agrupamos los resultados por el nombre de la empresa y el género de los empleados, lo cual nos permite calcular el salario promedio para cada combinación de empresa y género.

```sql
SELECT EM.nombre_empresa, E.género, AVG(E.salario) AS salario_promedio
FROM Empleados AS E
JOIN Sucursales AS S ON E.sucursal_id = S.id_sucursal
JOIN Empresas AS EM ON S.empresa_id = EM.id_empresa
GROUP BY EM.nombre_empresa, E.género;
```

nombre_empresa	genero	salario_promedio
ecopetrol	Masculino	1003500.000000
arturo calle	Femenino	60000.000000
arturo calle	masculino	70000.000000
exito	Femenino	88000.000000
exito	masculino	100000.000000
movistar	Femenino	6000.000000
terpel	Femenino	99000.000000

CONSULTA 39

Vamos a imaginar que cada departamento es como un hábitat diferente donde los dinosaurios trabajan, y que el salario de los empleados es como la cantidad de recursos que los dinosaurios obtienen en su hábitat.

En la tabla "Empleados", cada registro representa a un empleado, que sería como un dinosaurio individual. La columna "salario" representa el salario de cada empleado, que en este caso, sería como la cantidad de recursos que los dinosaurios obtienen en su hábitat.

En la tabla "Departamentos", cada registro representa un departamento de la empresa, que se puede considerar como diferentes hábitats donde trabajan los dinosaurios.

Al utilizar la cláusula JOIN, unimos la tabla "Empleados" con la tabla "Departamentos" utilizando el campo "departamento_id", de manera similar a cómo los dinosaurios pueden pertenecer y trabajar en diferentes departamentos según su especialización.

Utilizando la cláusula WHERE E.salario > 50000, filtramos los empleados cuyos salarios son mayores a 50000, lo cual nos permite seleccionar a aquellos dinosaurios que obtienen una mayor cantidad de recursos en su hábitat.

Utilizando la cláusula GROUP BY D.nombre_departamento, agrupamos los resultados por el nombre del departamento, lo cual nos permite contar el número de empleados que cumplen con la condición establecida en cada departamento.

```sql
SELECT D.nombre_departamento, COUNT(*) AS numero_empleados
FROM Empleados AS E
JOIN Departamentos AS D ON E.departamento_id = D.id_departamento
WHERE E.salario > 50000
GROUP BY D.nombre_departamento;
```

nombre_departamento	numero_empleados
recursos	2
mantenimiento	2
ingenerios	1
Ventas	1

CONSULTA 40

Vamos a imaginar que cada sucursal es como un hábitat diferente donde los dinosaurios trabajan, y que el salario de los empleados es como la cantidad de recursos que los dinosaurios obtienen en su hábitat.

En la tabla "Empleados", cada registro representa a un empleado, que sería como un dinosaurio individual. La columna "salario" representa el salario de cada empleado, que en este caso, sería como la cantidad de recursos que los dinosaurios obtienen en su hábitat.

En la tabla "Sucursales", cada registro representa una sucursal de la empresa, que se puede considerar como diferentes hábitats donde trabajan los dinosaurios.

Al utilizar la cláusula JOIN, unimos la tabla "Empleados" con la tabla "Sucursales" utilizando el campo "sucursal_id", de manera similar a cómo los dinosaurios pueden pertenecer y trabajar en diferentes sucursales según su ubicación.

Utilizando la cláusula WHERE E.salario > 50000, filtramos los empleados cuyos salarios son mayores a 50000, lo cual nos permite seleccionar a aquellos dinosaurios que obtienen una mayor cantidad de recursos en su hábitat.

Utilizando la cláusula GROUP BY S.nombre_sucursal, agrupamos los resultados por el nombre de la sucursal, lo cual nos permite contar el número de empleados que cumplen con la condición establecida en cada sucursal.

```sql
SELECT S.nombre_sucursal, COUNT(*) AS numero_empleados
FROM Empleados AS E
JOIN Sucursales AS S ON E.sucursal_id = S.id_sucursal
WHERE E.salario > 50000
GROUP BY S.nombre_sucursal;
```

nombre_sucursal	numero_empleados
Sucursal B	2
Sucursal c	2
Sucursal e	1
Sucursal A	1

CONSULTA 41

Imaginemos que tenemos una tabla llamada "Empresas" que almacena información sobre diferentes empresas que operan en un parque temático de dinosaurios. Esta tabla contiene columnas como "nombre_empresa" y "id_empresa" que identifican cada empresa de manera única. Además, tenemos una tabla llamada "Sucursales" que registra información sobre las diferentes sucursales de estas empresas en el parque. La tabla "Sucursales" incluye la columna "empresa_id" que establece la relación con la tabla "Empresas". Por último, contamos con una tabla llamada "Empleados" que registra información sobre los empleados de cada sucursal. Esta tabla contiene una columna "salario" que indica el salario de cada empleado, y se relaciona con la tabla "Sucursales" a través de la columna "sucursal_id".

La consulta que mencionas tiene como objetivo obtener el nombre de las empresas y la cantidad de empleados de cada empresa que tienen un salario mayor a $50,000.

```sql
SELECT Empresas.nombre_empresa, COUNT(*) AS cantidad_empleados
FROM Empresas
JOIN Sucursales ON Empresas.id_empresa = Sucursales.empresa_id
JOIN Empleados ON Sucursales.id_sucursal = Empleados.sucursal_id
WHERE Empleados.salario > 50000.00
GROUP BY Empresas.id_empresa, Empresas.nombre_empresa;
```

	nombre_empresa	cantidad_empleados
▶	arturo calle	2
	exito	2
	terpel	1
	ecopetrol	1

CONSULTA 42

Supongamos que tenemos una tabla llamada "Departamentos" que registra información sobre los diferentes departamentos en un parque temático de dinosaurios. Esta tabla contiene la columna "nombre_departamento" que almacena el nombre de cada departamento y la columna "id_departamento" que los identifica de manera única. Además, tenemos una tabla llamada "Empleados" que contiene información sobre los empleados del parque. Esta tabla incluye una columna "departamento_id" que establece la relación con la tabla "Departamentos" a través de los identificadores correspondientes. También tenemos una tabla llamada "Aumentos_Salariales" que registra los aumentos de salario que se han otorgado a los empleados. Esta tabla incluye la columna "empleado_id" que establece la relación con la tabla "Empleados" y la columna "fecha_aumento" que registra la fecha en la que se realizó cada aumento salarial.

La consulta que mencionas tiene como objetivo obtener el nombre de cada departamento y la cantidad de empleados en cada uno de ellos que hayan recibido un aumento salarial en el último mes.

```sql
SELECT d.nombre_departamento, COUNT(e.id_empleado) AS cantidad_empleados
FROM Departamentos d
INNER JOIN Empleados e ON d.id_departamento = e.departamento_id
INNER JOIN Aumentos_Salariales a ON e.id_empleado = a.empleado_id
WHERE a.fecha_aumento >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
GROUP BY d.nombre_departamento;
```

	nombre_departamento	cantidad_empleados
▶	Ventas	1
	recursos	1
	mantenimiento	1
	soporte	1
	ingenerios	1

CONSULTA 43

Imaginemos que en el mundo de los dinosaurios existen diferentes sucursales encargadas de la crianza y cuidado de dinosaurios. Cada sucursal tiene varios dinoempleados y se registran aumentos salariales para algunos de ellos.

El objetivo de esta consulta es obtener información sobre las sucursales y el número de dinoempleados que han tenido aumentos salariales en los últimos 30 días.

Nos brinda el ID y el nombre de las sucursales, así como el número de dinoempleados que han tenido aumentos salariales en los últimos 30 días en cada sucursal. Esto nos permite identificar cuántos dinoempleados han sido beneficiados con aumentos salariales recientes en el mundo de los dinosaurios y realizar análisis comparativos entre las sucursales.

```sql
SELECT s.id_sucursal, s.nombre_sucursal, COUNT(DISTINCT a.empleado_id) AS num_empleados_aumento
FROM Sucursales s
JOIN Empleados e ON s.id_sucursal = e.sucursal_id
JOIN Aumentos_Salariales a ON e.id_empleado = a.empleado_id
WHERE a.fecha_aumento >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
GROUP BY s.id_sucursal, s.nombre_sucursal;
```

id_sucursal	nombre_sucursal	num_empleados_aumento
1	Sucursal A	1
2	Sucursal B	1
3	Sucursal c	1
4	Sucursal d	1
5	Sucursal e	1

CONSULTA 44

Imaginemos que en el mundo de los dinosaurios existen diferentes empresas que se dedican a la crianza y cuidado de dinosaurios. Cada empresa tiene varias sucursales, y en cada sucursal trabajan dinoempleados que pueden recibir aumentos salariales.

El objetivo de esta consulta es obtener información sobre las empresas y el número total de dinoempleados que han recibido aumentos salariales en los últimos 30 días.

Nos brinda el nombre de las empresas y el número total de dinoempleados que han recibido aumentos salariales en los últimos 30 días en cada empresa. Esto nos permite identificar cuántos dinoempleados han sido beneficiados con aumentos salariales recientes y realizar análisis comparativos entre las empresas del mundo de los dinosaurios.

```sql
SELECT e.nombre_empresa, COUNT(DISTINCT a.empleado_id) AS numero_empleados
FROM Empresas e
INNER JOIN Sucursales s ON e.id_empresa = s.empresa_id
INNER JOIN Empleados em ON s.id_sucursal = em.sucursal_id
INNER JOIN Aumentos_Salariales a ON em.id_empleado = a.empleado_id
WHERE a.fecha_aumento >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
GROUP BY e.nombre_empresa;
```

nombre_empresa	numero_empleados
arturo calle	1
ecopetrol	1
exito	1
movistar	1
terpel	1

CONSULTA 45

Supongamos que tenemos una tabla llamada "Empleados" que registra información sobre los empleados de un parque temático de dinosaurios. Esta tabla incluye columnas como "edad" (que indica la edad de cada empleado) y "departamento_id" (que establece la relación con la tabla "Departamentos" a través de los identificadores correspondientes). También tenemos una tabla llamada "Departamentos" que almacena información sobre los diferentes departamentos en el parque, y una tabla llamada "Pagos_Salarios" que registra los pagos de salarios realizados a los empleados. La tabla "Pagos_Salarios" incluye la columna "empleado_id" que establece la relación con la tabla "Empleados" y la columna "monto_pago" que indica el monto del pago salarial.

El objetivo de esta consulta es obtener el nombre de cada departamento, el rango de edad de los empleados y el salario promedio correspondiente a cada combinación de departamento y rango de edad.

```sql
SELECT d.nombre_departamento,
       CASE
           WHEN e.edad BETWEEN 18 AND 30 THEN '18-30'
           WHEN e.edad BETWEEN 31 AND 40 THEN '31-40'
           WHEN e.edad BETWEEN 41 AND 50 THEN '41-50'
           ELSE '51+'
       END AS rango_edad,
       AVG(p.monto_pago) AS salario_promedio
FROM Empleados e
JOIN Departamentos d ON e.departamento_id = d.id_departamento
JOIN Pagos_Salarios p ON e.id_empleado = p.empleado_id
GROUP BY d.nombre_departamento, rango_edad;
```

	nombre_departamento	rango_edad	salario_promedio
▶	Ventas	18-30	2500.000000
	recursos	18-30	3000.000000
	mantenimiento	18-30	3000.000000
	soporte	18-30	3000.000000
	ingenerios	18-30	3000.000000

CONSULTA 46

Supongamos que tenemos una tabla llamada "Empleados" que registra información sobre los empleados en un parque temático de dinosaurios. Esta tabla incluye una columna "edad" que indica la edad de cada empleado. También tenemos una tabla llamada "Pagos_Salarios" que registra los pagos de salarios realizados a los empleados. La tabla "Pagos_Salarios" incluye la columna "empleado_id" que establece la relación con la tabla "Empleados" y la columna "monto_pago" que indica el monto del pago salarial. Además, contamos con una tabla llamada "Sucursales" que almacena información sobre las diferentes sucursales en el parque, con la columna "nombre_sucursal" que indica el nombre de cada sucursal y la columna "id_sucursal" que las identifica de manera única.

El objetivo de esta consulta es obtener el nombre de cada sucursal, el rango de edad de los empleados y el salario promedio correspondiente a cada combinación de sucursal y rango de edad.

```sql
SELECT s.nombre_sucursal,
       CASE
           WHEN e.edad >= 18 AND e.edad <= 30 THEN '18-30'
           WHEN e.edad > 30 AND e.edad <= 40 THEN '31-40'
           WHEN e.edad > 40 AND e.edad <= 50 THEN '41-50'
           ELSE 'Más de 50'
       END AS rango_edad,
       AVG(ps.monto_pago) AS salario_promedio
FROM Empleados e
JOIN Pagos_Salarios ps ON e.id_empleado = ps.empleado_id
JOIN Sucursales s ON ps.sucursal_id = s.id_sucursal
GROUP BY s.nombre_sucursal, rango_edad;
```

nombre_sucursal	rango_edad	salario_promedio
Sucursal A	18-30	2500.000000
Sucursal B	18-30	3000.000000
Sucursal c	18-30	3000.000000
Sucursal d	18-30	3000.000000
Sucursal e	18-30	3000.000000

CONSULTA 47

Imaginemos que en el mundo de los dinosaurios existen diferentes empresas que tienen varias sucursales encargadas de la crianza y cuidado de dinosaurios. Cada sucursal cuenta con dinoempleados que reciben pagos salariales por su trabajo.

El objetivo de esta consulta es obtener información sobre las sucursales, departamentos, empresas y rangos de edad de los dinoempleados, junto con el salario promedio de cada grupo.

Nos brinda información sobre las sucursales, departamentos y empresas del mundo de los dinosaurios. Además, nos muestra el rango de edad de los dinoempleados y el salario promedio para cada grupo. Esto nos permite realizar análisis comparativos de salarios en diferentes rangos de edad en el contexto de las sucursales, departamentos y empresas.

```sql
SELECT e.sucursal_id, e.departamento_id, emp.nombre_empresa,
       CASE
         WHEN e.edad BETWEEN 18 AND 30 THEN '18-30'
         WHEN e.edad BETWEEN 31 AND 40 THEN '31-40'
         WHEN e.edad BETWEEN 41 AND 50 THEN '41-50'
         ELSE '51+'
       END AS age_range,
       AVG(ps.monto_pago) AS avg_salary
FROM Empleados e
INNER JOIN Pagos_Salarios ps ON e.id_empleado = ps.empleado_id
INNER JOIN Sucursales s ON e.sucursal_id = s.id_sucursal
INNER JOIN Empresas emp ON s.empresa_id = emp.id_empresa
WHERE e.salario > 0
GROUP BY e.sucursal_id, e.departamento_id, emp.nombre_empresa, age_range;
```

sucursal_id	departamento_id	nombre_empresa	age_range	avg_salary
1	1	ecopetrol	18-30	2500.000000
2	2	arturo calle	18-30	3000.000000
3	3	exito	18-30	3000.000000
4	4	movistar	18-30	3000.000000
5	5	terpel	18-30	3000.000000

CONSULTA 48

Imaginemos que en el mundo de los dinosaurios existen diferentes departamentos encargados de tareas específicas, como la cría de dinosaurios herbívoros, el cuidado de los dinosaurios carnívoros, la investigación paleontológica, etc. Y cada departamento cuenta con dinoempleados que reciben pagos salariales por su trabajo.

El objetivo de esta consulta es obtener información sobre los departamentos, los rangos de edad de los empleados y el salario promedio de cada grupo.

Nos brinda información sobre los departamentos del mundo de los dinosaurios. Además, nos muestra el rango de edad de los dinoempleados en cada departamento y el salario promedio para cada grupo. Esto nos permite analizar el salario promedio en diferentes rangos de edad dentro de cada departamento.

```sql
SELECT d.nombre_departamento,
       CASE
         WHEN e.edad >= 18 AND e.edad <= 30 THEN '18-30'
         WHEN e.edad > 30 AND e.edad <= 40 THEN '31-40'
         WHEN e.edad > 40 AND e.edad <= 50 THEN '41-50'
         ELSE '51+'
       END AS rango_edad,
       AVG(ps.monto_pago) AS salario_promedio
FROM Empleados e
JOIN Departamentos d ON e.departamento_id = d.id_departamento
JOIN Pagos_Salarios ps ON e.id_empleado = ps.empleado_id
GROUP BY d.nombre_departamento, rango_edad;
```

nombre_departamento	rango_edad	salario_promedio
Ventas	18-30	2500.000000
recursos	18-30	3000.000000
mantenimiento	18-30	3000.000000
soporte	18-30	3000.000000
ingenerios	18-30	3000.000000

CONSULTA 49

Supongamos que tenemos una tabla llamada "Empleados" que registra información sobre los empleados en un parque temático de dinosaurios. Esta tabla incluye una columna "edad" que indica la edad de cada empleado. También tenemos una tabla llamada "Pagos_Salarios" que registra los pagos de salarios realizados a los empleados. La tabla "Pagos_Salarios" incluye la columna "empleado_id" que establece la relación con la tabla "Empleados" y la columna "monto_pago" que indica el monto del pago salarial. Además, contamos con una tabla llamada "Sucursales" que almacena información sobre las diferentes sucursales en el parque, con la columna "nombre_sucursal" que indica el nombre de cada sucursal y la columna "id_sucursal" que las identifica de manera única.

El objetivo de esta consulta es obtener el nombre de cada sucursal, el rango de edad de los empleados y el salario promedio correspondiente a cada combinación de sucursal y rango de edad.

```sql
SELECT s.nombre_sucursal,
       CASE
         WHEN e.edad BETWEEN 18 AND 25 THEN '18-25'
         WHEN e.edad BETWEEN 26 AND 35 THEN '26-35'
         WHEN e.edad BETWEEN 36 AND 45 THEN '36-45'
         ELSE '46+'
       END AS rango_edad,
       AVG(ps.monto_pago) AS salario_promedio
FROM Empleados e
JOIN Pagos_Salarios ps ON e.id_empleado = ps.empleado_id
JOIN Sucursales s ON ps.sucursal_id = s.id_sucursal
GROUP BY s.nombre_sucursal, rango_edad;
```

	nombre_sucursal	rango_edad	salario_promedio
▶	Sucursal A	26-35	2500.000000
	Sucursal B	18-25	3000.000000
	Sucursal c	18-25	3000.000000
	Sucursal d	18-25	3000.000000
	Sucursal e	18-25	3000.000000

CONSULTA 50

Supongamos que tenemos una base de datos que registra información sobre los empleados de diferentes sucursales de empresas relacionadas con parques temáticos de dinosaurios. La tabla "Empleados" almacena datos como la edad de los empleados, la tabla "Pagos_Salarios" registra los pagos de salarios realizados a los empleados, la tabla "Sucursales" contiene información sobre las sucursales y la tabla "Empresas" almacena los datos de las diferentes empresas.

El objetivo de esta consulta es obtener el nombre de cada empresa, el rango de edad de los empleados y el salario promedio correspondiente a cada combinación de empresa y rango de edad.

```sql
SELECT Empresas.nombre_empresa,
       CASE
         WHEN Empleados.edad BETWEEN 20 AND 29 THEN '20-29'
         WHEN Empleados.edad BETWEEN 30 AND 39 THEN '30-39'
         WHEN Empleados.edad BETWEEN 40 AND 49 THEN '40-49'
         ELSE '50+'
       END AS rango_edad,
       AVG(Pagos_Salarios.monto_pago) AS salario_promedio
FROM Empleados
INNER JOIN Pagos_Salarios ON Empleados.id_empleado = Pagos_Salarios.empleado_id
INNER JOIN Sucursales ON Pagos_Salarios.sucursal_id = Sucursales.id_sucursal
INNER JOIN Empresas ON Sucursales.empresa_id = Empresas.id_empresa
GROUP BY Empresas.nombre_empresa, rango_edad;
```

nombre_empresa	rango_edad	salario_promedio
ecopetrol	30-39	2500.000000
arturo calle	20-29	3000.000000
exito	20-29	3000.000000
movistar	50+	3000.000000
terpel	20-29	3000.000000

NOMINA:

Imaginemos que hay un parque llamado "Parque DinoAventura" donde trabajan dos tipos de dinosaurios: los T-Rex y los Estegosaurios. Los T-Rex son los empleados y los Estegosaurios son los patrones. Vamos a calcular la nómina de los empleados y los patrones.

DEVENGADO

El devengado es el sueldo básico que los empleados reciben por trabajar. Supongamos que el sueldo básico de un T-Rex es de 100 huesos y el de un Estegosaurio es de 200 huesos.

Entonces, si hay 2 T-Rex en el parque, el devengado total de los empleados sería:

Devengado de los T-Rex = Sueldo básico de un T-Rex * Número de T-Rex

Devengado de los T-Rex = 100 huesos * 2 T-Rex

Devengado de los T-Rex = 200 huesos

PRESTACIONES

Las prestaciones son beneficios adicionales que se le otorgan a los empleados. En Colombia, se suele dar el equivalente a un mes de salario adicional llamado "prima" y las vacaciones equivalen a 15 días de salario.

Supongamos que el valor de la prima para los T-Rex es de 50 huesos y las vacaciones son de 30 huesos.

Entonces, las prestaciones de los T-Rex serían:

Prestaciones de los T-Rex = Prima + Vacaciones

Prestaciones de los T-Rex = 50 huesos + 30 huesos

Prestaciones de los T-Rex = 80 huesos

NOMINA:

EXTRAS

Los extras son pagos adicionales que los empleados pueden recibir por trabajar horas extras. Si un T-Rex trabaja 5 horas extras y cada hora extra se paga a 10 huesos, entonces los extras serían:
Extras de los T-Rex = Horas extras * Valor de la hora extra
Extras de los T-Rex = 5 horas extras * 10 huesos
Extras de los T-Rex = 50 huesos

DEDUCCIONES

Los extras son pagos adicionales que los empleados pueden recibir por trabajar horas extras. Si un T-Rex trabaja 5 horas extras y cada hora extra se paga a 10 huesos, entonces los extras serían:
Extras de los T-Rex = Horas extras * Valor de la hora extra Extras de los T-Rex = 5 horas extras * 10 huesos Extras de los T-Rex = 50 huesos

PARAFISCAL

El parafiscal es un aporte adicional que deben hacer los patrones por cada empleado. Supongamos que el parafiscal para cada T-Rex es de 30 huesos. Entonces, el parafiscal total sería:
Parafiscal total = Parafiscal por T-Rex * Número de T-Rex
Parafiscal total = 30 huesos * 2 T-Rex
Parafiscal total = 60 huesos

NOMINA:

Finalmente, para calcular la nómina de los empleados, sumamos el devengado, las prestaciones, los extras y restamos las deducciones:

Nómina de los T-Rex = Devengado de los T-Rex + Prestaciones de los T-Rex + Extras de los T-Rex - Deducciones de los T-Rex

Nómina de los T-Rex = 200 huesos + 80 huesos + 50 huesos - 20 huesos

Nómina de los T-Rex = 310 huesos

La nómina de los patrones se calcula sumando el devengado de los Estegosaurios y el parafiscal:

Nómina de los Estegosaurios = Devengado de los Estegosaurios + Parafiscal total

Nómina de los Estegosaurios = 200 huesos + 60 huesos

Nómina de los Estegosaurios = 260 huesos

¡Y eso es todo! Ahora tienes una idea de cómo se calcula la nómina en Colombia usando dinosaurios como ejemplo.

id_empleado	Cesantias	cesantias_interes	prima	vacaciones	TotalPrestaciones	diurna	nocturno	diurnaDominical	nocturnaDominical	recargo_nocturno	Total_Extras	Devengado	Salud
1	388.888889	2.592592592587	388.888889	194.444444	974.81481481258 7	145.83333333	0.00000000	291.666667	218.7500000	218.75000000	874.99999998	8784.99999998	314.9999999992
2	1666.666667	5.555555555553	1666.666667	833.333333	4172.222222220553	1562.50000000	875.00000000	1500.000000	2500.0000000	1562.50000000	8000.00000000	75800.00000000	2720.0000000000
3	1711.111111	3.992592592592	1711.111111	855.555556	4281.770370369592	2750.00000000	4491.66666666	5133.333333	916.6666667	916.66666667	14208.33333331	113648.33333331	4088.3333333323
4	250.000000	1.250000000000	250.000000	125.000000	626.250000000000	218.75000000	350.00000000	300.000000	125.0000000	125.00000000	1118.75000000	7898.75000000	284.7500000000

Pension	FondoSol	uvt	UVT_Real	ReFuente	Deducciones	Neto	Nomina_Empleados	Salud_Jefe	Pension_Jefe	arl	Sena	ICBF
314.9999999992	118.12499999970	130	5513560	1047576.40	1048324.524999998 10	-1039539.52500001810	-1038564.710185205513	8707.64999998	944.9999999976	4.11074999998956	157.4999999996	236.2499999994
2720.0000000000	1020.00000000000	120	5089440	966993.60	973453.60000000000	-897653.60000000000	-893481.377777779447	75137.00000000	8160.0000000000	35.49600000000000	1360.0000000000	2040.0000000000
4088.3333333323	1533.12499999961	130	5513560	1047576.40	1057286.19166666421	-943637.85833335671	-939356.087962987120	112675.93333331	12264.9999999969	53.35274999998652	2044.1666666662	3066.2499999992
284.7500000000	106.78125000000	140	5937680	1128159.20	1128835.48125000000	-1120936.73125000000	-1120310.481250000000	7832.45000000	854.2500000000	3.71598750000000	142.3750000000	213.5625000000
4393.1250000000	1647.42187500000	130	5513560	1047576.40	1058010.07187500000	-935311.94687500000	-927738.355208333333	121604.17500000	13179.3750000000	57.33028125000000	2196.5625000000	3294.8437500000

caja_compe	TotalParafiscal	Nomina_Empresario
314.9999999992	52441.0749798800	-986123.635205325513
2720.0000000000	452494.9284000000	-440986.449377779447
4088.3333333323	678509.2283198450	-260846.859643142120
284.7500000000	47161.9928400000	-1073148.488410000000
4393.1250000000	732385.3818600000	-195352.973348333333

VIDEO

MIRA EL MEJOR VIDEO

With

https://www.youtube.com/watch?v=iHm9FpdWdnc

PRESENTACION VIDEO

Introducción a SQL Workbench

EDWIN ALBORNOZ
JERONIMO ESPITIA
JUAN DIEGO RODRIGUEZ
JUAN JOSE OROZCO
JUAN DAVID VALENCIA

¿Qué es SQL Workbench?

SQL Workbench es una herramienta de administración de bases de datos, que permite crear, modificar y administrar bases de datos SQL con facilidad.

Características principales

Permite un almacenamiento y una recuperación eficientes de los datos.

Proporciona herramientas para el análisis y la gestión de datos.

Admite múltiples usuarios y funciones de seguridad para controlar el acceso a los datos.

¿Cómo conectar una base de datos?

Para conectar una base de datos en SQL Workbench, se debe ingresar el nombre del servidor, puerto, nombre de usuario y contraseña en la pestaña 'Gestor de Conexiones'.

Gestión de la base de datos

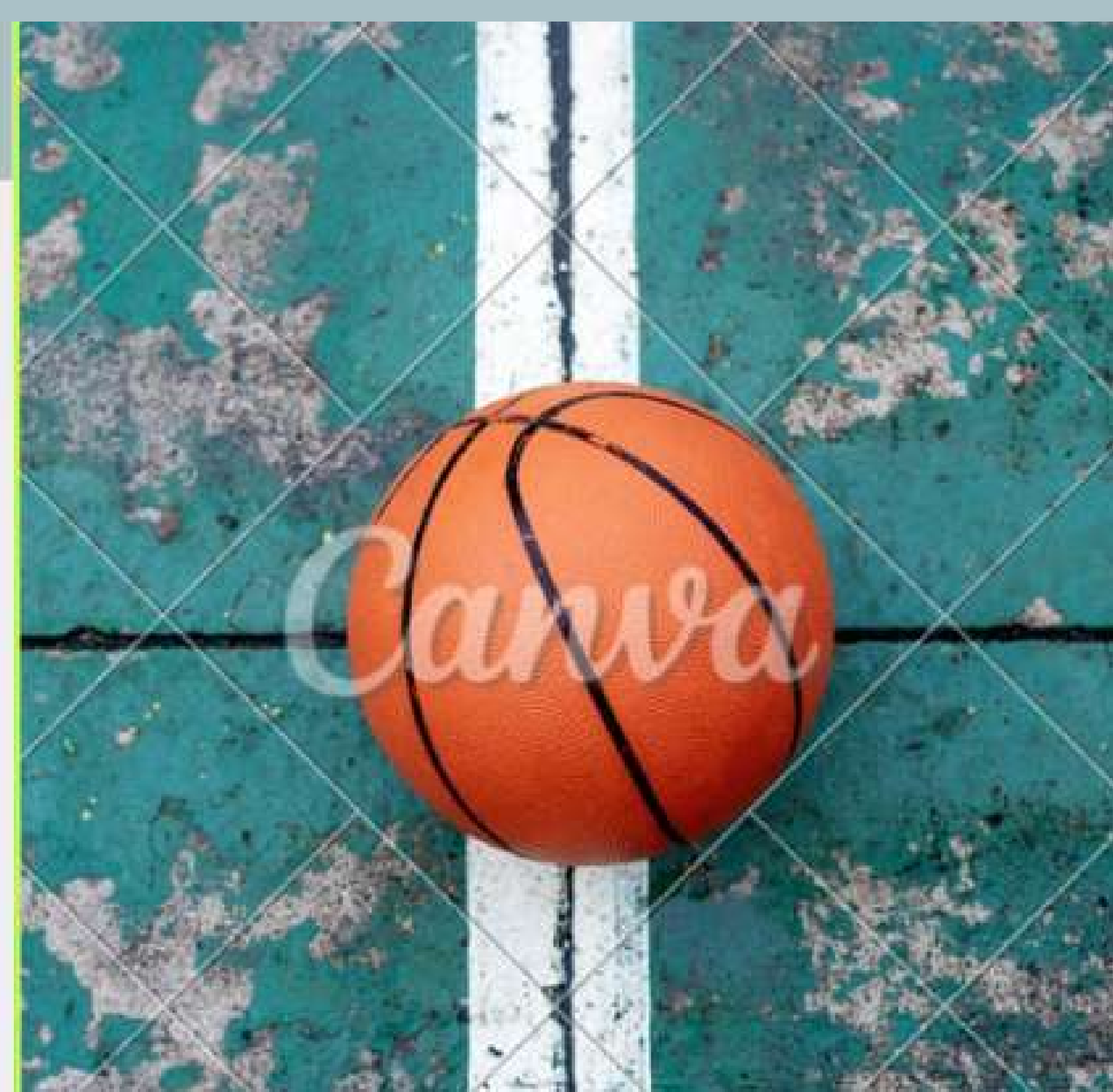
SQL Workbench es una herramienta de administración de bases de datos de código abierto que permite conectarse a distintos tipos de bases de datos y trabajar con ellas.

Exportación de datos

Para exportar una base de datos en SQL Workbench, selecciona la base de datos y haz clic en 'Data Export'. Luego, elige el formato y guarda el archivo.

Conclusiones y recomendaciones

Para mejorar su base de datos de SQL Workbench, considere optimizar consultas, indexar tablas, mejorar la calidad de los datos e implementar copias de seguridad periódicas.



MUCHAS GRACIAS
POR SU ATENCIÓN