PRACTICA 2 : INTRODUCCIÓN A BASES DE DATOS

1. Crear la Tabla `empleados`

```
CREATE TABLE empleados (
  id INTEGER AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  salario DECIMAL(10, 2) NOT NULL,
  departamento VARCHAR(50) NOT NULL,
  fecha_contratacion DATE NOT NULL,
  jefe_id INT,
  FOREIGN KEY (jefe_id) REFERENCES empleados(id)
);
```

2. Insertar Datos en la Tabla `empleados`

```
INSERT INTO empleados (nombre, salario, departamento, fecha_contratacion, jefe_id)
VALUES
('Juan Pérez', 55000, 'Ventas', '2021-01-01', NULL),
('María López', 45000, 'Marketing', '2020-05-15', NULL),
('Carlos Gómez', 60000, 'Ingeniería', '2019-08-20', NULL),
('Ana Rodríguez', 35000, 'Recursos Humanos', '2022-03-10', NULL),
('Luis Martínez', 50000, 'Ventas', '2021-06-01', 1),
('Laura García', 40000, 'Marketing', '2020-11-20', 2),
('Pedro Sánchez', 65000, 'Ingeniería', '2018-07-15', 3),
('Sofía Fernández', 30000, 'Recursos Humanos', '2022-09-05', 4);
```

3. Seleccionar Datos con Condiciones Complejas

```
SELECT *
FROM empleados
WHERE salario > 50000 AND departamento = 'Ventas' OR (salario < 30000 AND departamento =
'Marketing');
```

PRACTICA 2 : INTRODUCCIÓN A BASES DE DATOS

4. Agrupar y Agregar Datos

```
SELECT departamento, AVG(salario) AS salario_promedio
FROM empleados
GROUP BY departamento
HAVING AVG(salario) > 40000;
```

5. Ordenar Datos

```
SELECT *
FROM empleados
ORDER BY salario DESC, nombre ASC;
```

6. Actualizar Datos con Condiciones

```
UPDATE empleados
SET salario = salario * 1.10
WHERE departamento = 'Ventas' AND salario < 50000;
```

7. Eliminar Datos con Condiciones

```
DELETE FROM empleados
WHERE fecha_contratacion < '2020-01-01' AND departamento = 'Recursos Humanos';
```

8. Unir Datos con Subconsultas

```
SELECT e1.nombre, e1.salario
FROM empleados e1
WHERE e1.salario > (SELECT AVG(salario) FROM empleados);
```

9. Usar Funciones de Ventana

```
SELECT nombre, salario,
    RANK() OVER (ORDER BY salario DESC) AS ranking
FROM empleados;
```

10. Crear Vistas

```
CREATE VIEW empleados_ventas AS
SELECT nombre, salario
FROM empleados
WHERE departamento = 'Ventas';
```

11. Usar Subconsultas Correlacionadas

```
SELECT nombre, salario
FROM empleados e1
WHERE salario > (SELECT AVG(salario) FROM empleados e2 WHERE e1.departamento = e2.departamento);
```

12. Usar Índices

```
CREATE INDEX idx_departamento ON empleados (departamento);
```

Ejercicios para practicar y entregar

1. Crear la Tabla `productos`

```
CREATE TABLE productos (
    id INTEGER AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    precio DECIMAL(10, 2) NOT NULL,
    categoria VARCHAR(50) NOT NULL,
    fecha_lanzamiento DATE NOT NULL,
    proveedor_id INT,
```

FOREIGN KEY (proveedor_id) REFERENCES proveedores(id)

);

2. Insertar Datos en la Tabla `productos`

INSERT INTO productos (nombre, precio, categoria, fecha_lanzamiento, proveedor_id)

VALUES

('Laptop', 1200.00, 'Electrónica', '2021-01-01', NULL),

('Smartphone', 800.00, 'Electrónica', '2020-05-15', NULL),

('Cafetera', 50.00, 'Hogar', '2019-08-20', NULL),

('Libro', 20.00, 'Libros', '2022-03-10', NULL),

('Tablet', 300.00, 'Electrónica', '2021-06-01', 1),

('Aspiradora', 150.00, 'Hogar', '2020-11-20', 2),

('Monitor', 250.00, 'Electrónica', '2018-07-15', 3),

('Juego de Mesa', 30.00, 'Juegos', '2022-09-05', 4);

3. Seleccionar Datos con Condiciones Complejas

Escribe una consulta SQL que seleccione todos los productos cuyo precio sea mayor a 200.00 y pertenezcan a la categoría 'Electrónica', o cuyo precio sea menor a 50.00 y pertenezcan a la categoría 'Hogar'.

4. Agrupar y Agregar Datos

Escribe una consulta SQL que calcule el precio promedio de los productos agrupados por categoría y que solo muestre las categorías cuyo precio promedio sea mayor a 100.00.

5. Ordenar Datos

Escribe una consulta SQL que seleccione todos los productos y los ordene en orden descendente por precio y en orden ascendente por nombre.

6. Actualizar Datos con Condiciones

Escribe una consulta SQL que actualice el precio de todos los productos de la categoría 'Electrónica' que fueron lanzados después del 1 de enero de 2020, incrementando su precio en un 10%.

7. Eliminar Datos con Condiciones

Escribe una consulta SQL que elimine todos los productos de la categoría 'Libros' que fueron lanzados antes del 1 de enero de 2020.

8. Unir Datos con Subconsultas

Escribe una consulta SQL que seleccione el nombre y el precio de los productos cuyo precio sea mayor que el promedio de precios de todos los productos.

9. Usar Funciones de Ventana

Escribe una consulta SQL que seleccione el nombre y el precio de los productos y les asigne un ranking basado en el precio en orden descendente.

10. Crear Vistas

Escribe una consulta SQL que cree una vista llamada `productos_electronica` que seleccione el nombre y el precio de los productos de la categoría 'Electrónica'.

11. Usar Subconsultas Correlacionadas

Escribe una consulta SQL que seleccione el nombre y el precio de los productos cuyo precio sea mayor que el promedio de precios de los productos de la misma categoría.

12. Usar Índices

13. Seleccionar Productos con Precio Mayor al Promedio

Escribe una consulta SQL que seleccione el nombre y el precio de los productos cuyo precio sea mayor que el promedio de precios de todos los productos.

14. Seleccionar Productos con Precio Mayor al Promedio de su Categoría

Escribe una consulta SQL que seleccione el nombre y el precio de los productos cuyo precio sea mayor que el promedio de precios de los productos de la misma categoría.

15. Seleccionar Productos con Precio Mayor al Mínimo de su Categoría

Escribe una consulta SQL que seleccione el nombre y el precio de los productos cuyo precio sea mayor que el precio mínimo de los productos de la misma categoría.

16. Seleccionar Productos con Precio Mayor al Máximo de su Categoría

Escribe una consulta SQL que seleccione el nombre y el precio de los productos cuyo precio sea mayor que el precio máximo de los productos de la misma categoría.

17. Seleccionar Productos con Precio Igual al Máximo de su Categoría

Escribe una consulta SQL que seleccione el nombre y el precio de los productos cuyo precio sea igual al precio máximo de los productos de la misma categoría.

18. Seleccionar Productos con Precio Igual al Mínimo de su Categoría

Escribe una consulta SQL que seleccione el nombre y el precio de los productos cuyo precio sea igual al precio mínimo de los productos de la misma categoría.

19. Seleccionar Productos con Precio Mayor al Promedio de Todos los Productos

Escribe una consulta SQL que seleccione el nombre y el precio de los productos cuyo precio sea mayor que el promedio de precios de todos los productos en la tabla.

20. Seleccionar Productos con Precio Mayor al Promedio de Productos de Otra Categoría

Escribe una consulta SQL que seleccione el nombre y el precio de los productos cuyo precio sea mayor que el promedio de precios de los productos de una categoría específica (por ejemplo, 'Electrónica').

21. Seleccionar Productos con Precio Mayor al Promedio de Productos de la Misma Categoría y Fecha de Lanzamiento Reciente

Escribe una consulta SQL que seleccione el nombre y el precio de los productos cuyo precio sea mayor que el promedio de precios de los productos de la misma categoría y que hayan sido lanzados después de una fecha específica (por ejemplo, '2020-01-01').

22. Seleccionar Productos con Precio Mayor al Promedio de Productos de la Misma Categoría y Proveedor Asignado

Escribe una consulta SQL que seleccione el nombre y el precio de los productos cuyo precio sea mayor que el promedio de precios de los productos de la misma categoría y que tengan un proveedor asignado (proveedor_id no es NULL).