Ejercicios interesantes

Base de datos SQL

1. Visualizar el departamento con más personal del oficio 'empleado'.

```
SELECT DNOMBRE, COUNT(*) AS 'EMPLEADOS'
FROM VENTAS.EMPLEADOS E
    INNER JOIN VENTAS.DEPARTAMENTOS D ON E.DEP_NO = D.DEP_NO
WHERE
    OFICIO = 'EMPLEADO'
GROUP BY
    DNOMBRE
ORDER BY 2 DESC
LIMIT 1;

SELECT DNOMBRE, COUNT(*) AS 'EMPLEADOS'
FROM VENTAS.EMPLEADOS E, VENTAS.DEPARTAMENTOS D
WHERE
    E.DEP_NO = D.DEP_NO
    AND OFICIO = 'EMPLEADO'
GROUP BY
    DNOMBRE
ORDER BY 2 DESC
LIMIT 1;
```

Código SQL

2. Visualizar el departamento con más presupuesto asignado para pagar el salario y la comisión de sus empleados.

```
SELECT D.DEP_NO, D.DNOMBRE, (
        SUM(E.SALARIO) + IFNULL(E.COMISION, 0)
    ) AS 'SALARIOS'
FROM VENTAS.DEPARTAMENTOS D
    INNER JOIN VENTAS.EMPLEADOS E ON D.DEP_NO = E.DEP_NO
GROUP BY
    D.DEP_NO
ORDER BY 3 DESC
LIMIT 1;
```

Código SQL

3. Obtener información de los empleados que ganan más que cualquier empleado del departamento 30.

```
SELECT *
FROM VENTAS.EMPLEADOS
WHERE (SALARIO + IFNULL(COMISION, 0)) > (
        SELECT MAX(SALARIO + IFNULL(COMISION, 0))
        FROM VENTAS.EMPLEADOS
        GROUP BY
            DEP_NO
        HAVING
            DEP_NO = 30
    );

SELECT *
FROM VENTAS.EMPLEADOS
WHERE (SALARIO + IFNULL(COMISION, 0)) > ALL (
        SELECT SALARIO + IFNULL(COMISION, 0)
        FROM VENTAS.EMPLEADOS
        WHERE
            DEP_NO = 30
    );

SELECT *
FROM VENTAS.EMPLEADOS
WHERE
    SALARIO + IFNULL(COMISION, 0) > (
        SELECT MAX(SALARIO + IFNULL(COMISION, 0))
        FROM VENTAS.EMPLEADOS
        WHERE
            DEP_NO = 30
    );
```

Código SQL

4. Obtén los empleados cuyo salario supera al de sus compañeros de departamento.

```
SELECT *
FROM VENTAS.EMPLEADOS E
WHERE
    SALARIO > (
        SELECT MAX(SALARIO)
        FROM VENTAS.EMPLEADOS E2
        WHERE
            E.DEP_NO = E2.DEP_NO
            AND E.EMP_NO != E2.EMP_NO
    );
```

Código SQL

5. Obtén un listado de los productos vendidos por cada empleado (apellido del empleado y descripción y precio actual de los productos).

```sql
-- Consulta multitabla
SELECT DISTINCT
    E.APELLIDO,
    PR.DESCRIPCION,
    PR.PRECIO_ACTUAL
FROM VENTAS.PRODUCTOS PR, VENTAS.PEDIDOS P, VENTAS.CLIENTES C, VENTAS.EMPLEADOS E
WHERE
    PR.PRODUCTO_NO = P.PRODUCTO_NO
    AND P.CLIENTE_NO = C.CLIENTE_NO
    AND C.VENDEDOR_NO = E.EMP_NO;

-- INNER JOIN
SELECT DISTINCT
    E.APELLIDO,
    PR.DESCRIPCION,
    PR.PRECIO_ACTUAL
FROM VENTAS.PRODUCTOS PR
    INNER JOIN VENTAS.PEDIDOS P ON PR.PRODUCTO_NO = P.PRODUCTO_NO
    INNER JOIN VENTAS.CLIENTES C ON P.CLIENTE_NO = C.CLIENTE_NO
    INNER JOIN VENTAS.EMPLEADOS E ON C.VENDEDOR_NO = E.EMP_NO;
```

Código SQL

6. Indicar el nombre y la fecha en que se ha vendido por última vez cada producto siempre y cuando el importe total de ventas de ese producto sea superior a 100.

```sql
-- Consulta multitabla
SELECT PR.DESCRIPCION, MAX(P.FECHA_PEDIDO) AS 'FECHA DE LA ÚLTIMA VENTA'
FROM VENTAS.PEDIDOS P, VENTAS.PRODUCTOS PR
WHERE
    P.PRODUCTO_NO = PR.PRODUCTO_NO
GROUP BY
    PR.DESCRIPCION
HAVING
    SUM(PR.PRECIO_ACTUAL * P.UNIDADES) > 100;

-- INNER JOIN
SELECT PR.DESCRIPCION, MAX(P.FECHA_PEDIDO) AS 'FECHA DE LA ÚLTIMA VENTA'
FROM VENTAS.PEDIDOS P
    INNER JOIN VENTAS.PRODUCTOS PR ON P.PRODUCTO_NO = PR.PRODUCTO_NO
GROUP BY
    PR.DESCRIPCION
HAVING
    SUM(PR.PRECIO_ACTUAL * P.UNIDADES) > 100;
```

Código SQL

7. Listar las localidades donde existan departamentos con empleados cuya comisión supere el 10 % del salario.

```sql
-- Consulta multitabla
SELECT DISTINCT
    D.LOCALIDAD
FROM VENTAS.DEPARTAMENTOS D, VENTAS.EMPLEADOS E
WHERE
    D.DEP_NO = E.DEP_NO
    AND E.COMISION > E.SALARIO * 0.1;

-- INNER JOIN
SELECT DISTINCT
    D.LOCALIDAD
FROM VENTAS.DEPARTAMENTOS D
    INNER JOIN VENTAS.EMPLEADOS E ON D.DEP_NO = E.DEP_NO
WHERE
    E.COMISION > E.SALARIO * 0.1;
```

Código SQL

8. Obtener el nombre de cada departamento y la media salarial de ese departamento, para los departamentos cuya media salarial es mayor que el salario del empleado 7654.

```sql
-- Consulta multitabla
SELECT D.DNOMBRE, AVG(SALARIO) AS 'MEDIA SALARIAL'
FROM VENTAS.DEPARTAMENTOS D, VENTAS.EMPLEADOS E
WHERE
    D.DEP_NO = E.DEP_NO
GROUP BY
    D.DEP_NO
HAVING
    AVG(SALARIO) > (
        SELECT SALARIO
        FROM VENTAS.EMPLEADOS
        WHERE
            EMP_NO = '7654'
    );

-- INNER JOIN
SELECT D.DNOMBRE, AVG(SALARIO) AS 'MEDIA SALARIAL'
FROM VENTAS.DEPARTAMENTOS D
    INNER JOIN VENTAS.EMPLEADOS E ON D.DEP_NO = E.DEP_NO
GROUP BY
    D.DEP_NO
HAVING
    AVG(SALARIO) > (
        SELECT SALARIO
        FROM VENTAS.EMPLEADOS
        WHERE
            EMP_NO = '7654'
    );
```

Código SQL

9. Inserta con los códigos 8000, 8001 y 8002 a los empleados “MARTINEZ”, “PEREZ” y “RODRIGUEZ”, analistas que acaban de ser contratados por el departamento de operaciones. Sus salarios serán respectivamente de 1000, 1100 y 1200€, sin comisiones. Su director será el 7521.

```
-- Especificando el orden de valores
INSERT INTO
    VENTAS.EMPLEADOS (
        EMP_NO, APELLIDO, OFICIO, DIRECTOR, FECHA_ALTA, SALARIO, COMISION, DEP_NO,
TELEFONO
    )
VALUES (
        8000, 'MARTINEZ', 'ANALISTAS', '7521', '2024-04-09', 1000, NULL, '50',
666777888
    ),
    (
        8001, 'PEREZ', 'ANALISTAS', '7521', '2024-04-09', 1100, NULL, '50',
666777888
    ),
    (
        8002, 'RODRIGUEZ', 'ANALISTAS', '7521', '2024-04-09', 1200, NULL, '50',
666777888
    );

-- Sin especificar el orden de los valores
INSERT INTO
    VENTAS.EMPLEADOS
VALUES (
        8000, 'MARTINEZ', 'ANALISTAS', '7521', '2024-04-09', 1000, NULL, '50',
666777888
    ),
    (
        8001, 'PEREZ', 'ANALISTAS', '7521', '2024-04-09', 1100, NULL, '50',
666777888
    ),
    (
        8002, 'RODRIGUEZ', 'ANALISTAS', '7521', '2024-04-09', 1200, NULL, '50',
666777888
    );
```

Código SQL

10. El pedido 1001 no ha sido pagado, por ese motivo debemos añadir al debe del cliente el importe del pedido.

```
-- Consulta multitabla
UPDATE VENTAS.PRODUCTOS PR, VENTAS.PEDIDOS P, VENTAS.CLIENTES C
SET
    C.DEBE = C.DEBE + (PR.PRECIO_ACTUAL * P.UNIDADES)
```

```
WHERE
    P.PEDIDO_NO = 1001
    AND PR.PRODUCTO_NO = P.PRODUCTO_NO
    AND P.CLIENTE_NO = C.CLIENTE_NO;

-- INNER JOIN
UPDATE VENTAS.PEDIDOS P
INNER JOIN VENTAS.CLIENTES C ON P.CLIENTE_NO = C.CLIENTE_NO
INNER JOIN VENTAS.PRODUCTOS PR ON P.PRODUCTO_NO = PR.PRODUCTO_NO
SET
    C.DEBE = C.DEBE + (P.UNIDADES * PR.PRECIO_ACTUAL)
WHERE
    P.PEDIDO_NO = 1001;

SELECT * FROM VENTAS.CLIENTES;
```

Código SQL

11. Copia todos los empleados entre el 7839 y el 7876, poniéndoles como código el código actual menos 7838. Su fecha de contratación debe ser la fecha actual.

```
INSERT INTO
    VENTAS.EMPLEADOS (
        EMP_NO, APELLIDO, OFICIO, DIRECTOR, FECHA_ALTA, SALARIO, COMISION, DEP_NO,
TELEFONO
    )
SELECT
    EMP_NO - 7838,
    APELLIDO,
    OFICIO,
    DIRECTOR,
    FECHA_ALTA,
    SALARIO,
    COMISION,
    DEP_NO,
    TELEFONO
FROM VENTAS.EMPLEADOS
WHERE
    EMP_NO BETWEEN 7839 AND 7876;
```

Código SQL

12. Crea una nueva tabla de “EMP_CONT” con la misma estructura que la tabla de empleados, e inserta en ella los empleados del departamento de contabilidad.

```
-- Varias consultas
CREATE TABLE VENTAS.EMP_CONT (
    EMP_NO SMALLINT(4) NOT NULL, APELLIDO VARCHAR(20), OFICIO VARCHAR(20),
DIRECTOR SMALLINT(4), FECHA_ALTA DATE, SALARIO FLOAT(8, 2), COMISION FLOAT(8, 2),
```

```
DEP_NO SMALLINT(4), TELEFONO VARCHAR(15), PRIMARY KEY (EMP_NO), INDEX
FK_EMP_DIRECTOR (DIRECTOR ASC), INDEX FK_EMP_CONT_DEP_NO (DEP_NO ASC), CONSTRAINT
FK_EMP_CONT_DEP_NO FOREIGN KEY (DEP_NO) REFERENCES DEPARTAMENTOS (DEP_NO) ON
UPDATE CASCADE
);

INSERT INTO
    VENTAS.EMP_CONT
SELECT *
FROM VENTAS.EMPLEADOS
WHERE
    DEP_NO = (
        SELECT DEP_NO
        FROM VENTAS.DEPARTAMENTOS
        WHERE
            DNOMBRE = 'CONTABILIDAD'
    );

-- Una consulta
CREATE TABLE EMP_CONT AS
SELECT *
FROM VENTAS.EMPLEADOS
WHERE
    DEP_NO = (
        SELECT DEP_NO
        FROM VENTAS.DEPARTAMENTOS
        WHERE
            DNOMBRE = 'CONTABILIDAD'
    );
```

Código SQL

13. Sube el salario un 2 % a los empleados con oficio vendedor y súbelo un 3 % a los que no son vendedor ni director (con una sola instrucción)

```
UPDATE VENTAS.EMPLEADOS
SET
    SALARIO = SALARIO * (
        CASE
            WHEN OFICIO = 'VENDEDOR' THEN 1.02
            WHEN OFICIO NOT IN('VENDEDOR', 'DIRECTOR') THEN 1.03
            ELSE 1
        END
    );
```

Código SQL

14. Crea una nueva tabla PUESTOS(codpuesto,puesto). Inserta en dicha tabla los distintos puestos que ocupan los empleados (sin repetir).

```
CREATE TABLE VENTAS.PUESTOS (
    CODPUESTO SMALLINT AUTO_INCREMENT PRIMARY KEY, PUESTO VARCHAR(20)
);

INSERT INTO
    VENTAS.PUESTOS (PUESTO)
SELECT DISTINCT
    OFICIO
FROM VENTAS.EMPLEADOS;
```

Código SQL

15. /* Crea un campo codpuesto en la tabla de empleado y asigna a cada empleado el código de puesto que le corresponda en función de su oficio. Elimina posteriormente el campo OFICIO de la tabla de empleados.

```
ALTER TABLE VENTAS.EMPLEADOS ADD CODPUESTO SMALLINT;

UPDATE VENTAS.EMPLEADOS E
SET
    E.CODPUESTO = (
        SELECT P.CODPUESTO
        FROM VENTAS.PUESTOS P
        WHERE
            E.OFICIO = P.PUESTO
    );

ALTER TABLE VENTAS.EMPLEADOS DROP COLUMN OFICIO;
```

Código SQL

16. Borra el departamento CONTABILIDAD. Antes de eliminarlo, tendrás que reubicar a sus empleados y asignarlos al departamento DECONOCIDO.

```
UPDATE VENTAS.EMPLEADOS
SET
    DEP_NO = (
        SELECT DEP_NO
        FROM VENTAS.DEPARTAMENTOS
        WHERE
            DNOMBRE = 'DESCONOCIDO'
    )
WHERE
    DEP_NO IN (
        SELECT DEP_NO
        FROM VENTAS.DEPARTAMENTOS
        WHERE
            DNOMBRE = 'CONTABILIDAD'
```

```
    );

DELETE FROM VENTAS.DEPARTAMENTOS WHERE DNOMBRE = 'CONTABILIDAD';
```

Código SQL

17. Añade un campo SEDE de tipo smallint a la tabla de departamentos.

```
ALTER TABLE VENTAS.DEPARTAMENTOS ADD SEDE SMALLINT;
```

Código SQL

18. Asigna la sede 1 a todos los departamentos excepto al desconocido, que tendrá sede 0.

```
UPDATE VENTAS.DEPARTAMENTOS
SET
    SEDE = IF(
        DNOMBRE = 'CONTABILIDAD', 0, 1
    );
```

Código SQL

19. Duplica todos los departamentos de la sede 1 para crearlos también en la sede 2. Los códigos de los nuevos departamentos se obtendrán sumándole 50 al código de departamento que estén copiando.

```
INSERT INTO
    VENTAS.DEPARTAMENTOS (
        DEP_NO, DNOMBRE, LOCALIDAD, SEDE
    )
SELECT DEP_NO + 50, DNOMBRE, LOCALIDAD, 2
FROM VENTAS.DEPARTAMENTOS
WHERE
    SEDE = 1;
```

Código SQL