

Trabajo Práctico N° 4: SQL.

Ejercicio 1.

Cliente= (idCliente, nombre, apellido, DNI, telefono, direccion).

Factura= (nroTicket, total, fecha, hora, idCliente(Fk)).

Detalle= (nroTicket(Fk), idProducto(Fk), cantidad, precioUnitario).

Producto= (idProducto, nombreP, descripcion, precio, stock).

(1) Listar datos personales de clientes cuyo apellido comience con el string 'Pe'. Ordenar por DNI.

```
SELECT nombre, apellido, DNI, telefono, direccion
FROM Cliente
WHERE (apellido LIKE 'Pe%')
ORDER BY DNI
```

(2) Listar nombre, apellido, DNI, teléfono y dirección de clientes que realizaron compras sólo durante 2024.

```
SELECT c.nombre, c.apellido, c.DNI, c.telefono, c.direccion
FROM Factura f
NATURAL JOIN Cliente c
WHERE (YEAR(f.fecha) = 2024)
EXCEPT
SELECT c.nombre, c.apellido, c.DNI, c.telefono, c.direccion
FROM Factura f
NATURAL JOIN Cliente c
WHERE (YEAR(f.fecha) <> 2024)
```

(3) Listar nombre, descripción, precio y stock de productos vendidos al cliente con DNI 45.789.456, pero que no fueron vendidos a clientes de apellido 'Garcia'.

```
SELECT DISTINCT p.nombreP, p.descripcion, p.precio, p.stock
FROM Detalle d
INNER JOIN Factura f ON (d.nroTicket = f.nroTicket)
INNER JOIN Producto p ON (d.idProducto = p.idProducto)
INNER JOIN Cliente c ON (f.idCliente = c.idCliente)
WHERE (c.DNI = 45789456 AND p.idProducto NOT IN (
    SELECT d2.idProducto
    FROM Detalle d2
    INNER JOIN Factura f2 ON (d2.nroTicket = f2.nroTicket)
    INNER JOIN Cliente c2 ON (f2.idCliente = c2.idCliente)
))
```

```
        WHERE (c2.apellido = 'Garcia')  
))
```

- (4)** Listar nombre, descripción, precio y stock de productos no vendidos a clientes que tengan teléfono con característica 221 (la característica está al comienzo del teléfono). Ordenar por nombre.

```
SELECT p.nombreP, p.descripcion, p.precio, p.stock  
FROM Producto p  
WHERE (p.idProducto NOT IN (  
    SELECT p.idProducto  
    FROM Detalle d  
    INNER JOIN Factura f ON (d.nroTicket = f.nroTicket)  
    INNER JOIN Producto p ON (d.idProducto = p.idProducto)  
    INNER JOIN Cliente c ON (f.idCliente = c.idCliente)  
    WHERE (c.telefono LIKE '221%')  
))  
ORDER BY p.nombreP
```

- (5)** Listar, para cada producto, nombre, descripción, precio y cuantas veces fue vendido. Tener en cuenta que puede no haberse vendido nunca el producto.

```
SELECT p.nombreP, p.descripcion, p.precio, SUM(COALESCE(d.cantidad, 0)) AS  
cantVendida  
FROM Detalle d  
RIGHT JOIN Producto p ON (d.idProducto = p.idProducto)  
GROUP BY p.idProducto, p.nombreP, p.descripcion, p.precio
```

- (6)** Listar nombre, apellido, DNI, teléfono y dirección de clientes que compraron los productos con nombre 'prod1' y 'prod2', pero nunca compraron el producto con nombre 'prod3'.

```
SELECT c.nombre, c.apellido, c.DNI, c.telefono, c.direccion  
FROM Detalle d  
INNER JOIN Factura f ON (d.nroTicket = f.nroTicket)  
INNER JOIN Producto p ON (d.idProducto = p.idProducto)  
INNER JOIN Cliente c ON (f.idCliente = c.idCliente)  
WHERE (p.nombreP IN ('prod1', 'prod2') AND c.idCliente NOT IN (  
    SELECT c.idCliente  
    FROM Detalle d  
    INNER JOIN Factura f ON (d.nroTicket = f.nroTicket)  
    INNER JOIN Producto p ON (d.idProducto = p.idProducto)  
    INNER JOIN Cliente c ON (f.idCliente = c.idCliente)  
    WHERE (p.nombreP = 'prod3')
```

```
)  
GROUP BY c.idCliente  
HAVING (COUNT(DISTINCT p.nombreP) = 2)
```

(7) Listar nroTicket, total, fecha, hora y DNI del cliente de aquellas facturas donde se haya comprado el producto 'prod38' o la factura tenga fecha de 2023.

```
SELECT DISTINCT f.nroTicket, f.total, f.fecha, f.hora, c.DNI  
FROM Detalle d  
INNER JOIN Factura f ON (d.nroTicket = f.nroTicket)  
INNER JOIN Producto p ON (d.idProducto = p.idProducto)  
INNER JOIN Cliente c ON (f.idCliente = c.idCliente)  
WHERE (YEAR(f.fecha) = 2023 OR p.nombreP = 'prod38')
```

(8) Agregar un cliente con los siguientes datos: nombre 'Jorge Luis', apellido 'Castor', DNI 40.578.999, teléfono '221-4400789', dirección '11 entre 500 y 501 Nro. 2587' e id de cliente 500.002. Se supone que el idCliente 500.002 no existe.

```
INSERT INTO Cliente (idCliente, nombre, apellido, DNI, telefono, direccion)  
VALUES (500002, 'Jorge Luis', 'Castor', 40578999, '221-4400789', '11 entre 500 y 501  
Nro. 2587')
```

(9) Listar nroTicket, total, fecha, hora para las facturas del cliente 'Jorge Pérez' donde no haya comprado el producto 'Z'.

```
SELECT f.nroTicket, f.total, f.fecha, f.hora  
FROM Factura f  
INNER JOIN Cliente c ON (f.idCliente = c.idCliente)  
WHERE (c.nombre = 'Jorge' AND c.apellido = 'Pérez') AND f.nroTicket NOT IN (  
    SELECT d.nroTicket  
    FROM Detalle d  
    INNER JOIN Producto p ON (d.idProducto = p.idProducto)  
    WHERE (p.nombreP = 'Z'))
```

(10) Listar DNI, apellido y nombre de clientes donde el monto total comprado, teniendo en cuenta todas sus facturas, supere \$100.000.

```
SELECT c.DNI, c.apellido, c.nombre  
FROM Factura f  
INNER JOIN Cliente c ON (f.idCliente = c.idCliente)
```

GROUP BY c.DNI, c.apellido, c.nombre
HAVING (SUM(f.total) > 100000)

Ejercicio 2.

Localidad = (codigoPostal, nombreL, descripcion, nroHabitantes).

Arbol = (nroArbol, especie, anios, calle, nro, codigoPostal(Fk)).

Podador = (DNI, nombre, apellido, telefono, fnac, codigoPostalVive(Fk)).

Poda = (codPoda, fecha, DNI(Fk), nroArbol(Fk)).

(1) Listar especie, años, calle, nro. y localidad de árboles podados por el podador ‘Juan Perez’ y por el podador ‘Jose Garcia’.

```
SELECT a.especie, a.anios, a.calle, a.nro, l.nombreL
FROM Arbol a
INNER JOIN Localidad l ON (a.codigoPostal = l.codigoPostal)
WHERE (EXISTS (
    SELECT 1
    FROM Poda p
    INNER JOIN Podador pod ON (p.DNI = pod.DNI)
    WHERE (a.nroArbol = p.nroArbol AND pod.nombre = 'Juan' AND pod.apellido
= 'Perez')
)
AND EXISTS (
    SELECT 1
    FROM Poda p
    INNER JOIN Podador pod ON (p.DNI = pod.DNI)
    WHERE (a.nroArbol = p.nroArbol AND pod.nombre = 'Jose' AND pod.apellido
= 'Garcia')
)
)
)
```

(2) Reportar DNI, nombre, apellido, fecha de nacimiento y localidad donde viven de aquellos podadores que tengan podas realizadas durante 2023.

```
SELECT DISTINCT pod.DNI, pod.nombre, pod.apellido, pod.telefono, pod.fnac,
l.nombreL
FROM Poda p
INNER JOIN Podador pod ON (p.DNI = pod.DNI)
INNER JOIN Localidad l ON (pod.codigoPostalVive = l.codigoPostal)
WHERE (YEAR(p.fecha) = 2023)
```

(3) Listar especie, años, calle, nro. y localidad de árboles que no fueron podados nunca.

```
SELECT a.especie, a.anios, a.calle, a.nro, l.nombreL
FROM Arbol a
INNER JOIN Localidad l ON (a.codigoPostal = l.codigoPostal)
WHERE (a.nroArbol NOT IN (
```

```
SELECT p.nroArbol
FROM Poda p
))
```

(4) Reportar especie, años, calle, nro. y localidad de árboles que fueron podados durante 2022 y no fueron podados durante 2023.

```
SELECT a.especie, a.anios, a.calle, a.nro, l.nombreL
FROM Arbol a
INNER JOIN Localidad l ON (a.codigoPostal = l.codigoPostal)
WHERE (a.nroArbol IN (
    SELECT p.nroArbol
    FROM Poda p
    WHERE (YEAR(p.fecha) = 2022)
))
AND a.nroArbol NOT IN (
    SELECT p.nroArbol
    FROM Poda p
    WHERE (YEAR(p.fecha) = 2023)
)
)
```

(5) Reportar DNI, nombre, apellido, fecha de nacimiento y localidad donde viven de aquellos podadores con apellido terminado con el string 'ata' y que tengan, al menos, una poda durante 2024. Ordenar por apellido y nombre.

```
SELECT pod.DNI, pod.nombre, pod.apellido, pod.fnac, l.nombreL
FROM Podador pod
INNER JOIN Localidad l ON (pod.codigoPostalVive = l.codigoPostal)
WHERE (pod.apellido LIKE '%ata' AND pod.DNI IN (
    SELECT p.DNI
    FROM Poda p
    WHERE (YEAR(p.fecha) = 2024)
))
ORDER BY pod.apellido, pod.nombre
```

(6) Listar DNI, apellido, nombre, teléfono y fecha de nacimiento de podadores que sólo podaron árboles de especie 'Conífera'.

```
SELECT pod.DNI, pod.apellido, pod.nombre, pod.telefono, pod.fnac
FROM Poda p
INNER JOIN Podador pod ON (p.DNI = pod.DNI)
INNER JOIN Arbol a ON (p.nroArbol = a.nroArbol)
WHERE (a.especie = 'Conífera')
```

EXCEPT

```
SELECT pod.DNI, pod.apellido, pod.nombre, pod.telefono, pod.fnac
FROM Poda p
INNER JOIN Podador pod ON (p.DNI = pod.DNI)
INNER JOIN Arbol a ON (p.nroArbol = a.nroArbol)
WHERE (a.especie <> 'Conífera')
```

(7) Listar especies de árboles que se encuentren en la localidad de 'La Plata' y también en la localidad de 'Salta'.

```
SELECT a.especie
FROM Arbol a
INNER JOIN Localidad l ON (a.codigoPostal = l.codigoPostal)
WHERE (l.nombreL = 'La Plata')
INTERSECT
SELECT a.especie
FROM Arbol a
INNER JOIN Localidad l ON (a.codigoPostal = l.codigoPostal)
WHERE (l.nombreL = 'Salta')
```

(8) Eliminar el podador con DNI 22.234.566.

```
DELETE FROM Poda WHERE (DNI = 22234566)
DELETE FROM Podador WHERE (DNI = 22234566)
```

(9) Reportar nombre, descripción y cantidad de habitantes de localidades que tengan menos de 5 árboles.

```
SELECT l.nombreL, l.descripcion, l.nroHabitantes
FROM Arbol a
RIGHT JOIN Localidad l ON (a.codigoPostal = l.codigoPostal)
GROUP BY l.codigoPostal, l.nombreL, l.descripcion, l.nroHabitantes
HAVING (COUNT(*) < 5)
```

Ejercicio 3.

Banda= (codigoB, nombreBanda, genero_musical, anio_creacion).
Integrante= (DNI, nombre, apellido, direccion, email, fecha_nacimiento, codigoB(Fk)).
Escenario= (nroEscenario, nombre_escenario, ubicacion, cubierto, m2, descripcion).
Recital= (fecha, hora, nroEscenario(Fk), codigoB(Fk)).

(1) Listar DNI, nombre, apellido, dirección y email de integrantes nacidos entre 1980 y 1990, y que hayan realizado algún recital durante 2023.

```
SELECT i.DNI, i.nombre, i.apellido, i.direccion, i.email
FROM Integrante i
WHERE (i.fecha_nacimiento BETWEEN '1980-01-01' AND '1990-12-31' AND i.DNI
IN (
    SELECT i.DNI
    FROM Recital r
    INNER JOIN Banda b ON (r.codigoB = b.codigoB)
    INNER JOIN Integrante i ON (b.codigoB = i.codigoB)
    WHERE (YEAR(r.fecha) = 2023)
))
```

(2) Reportar nombre, género musical y año de creación de bandas que hayan realizado recitales durante 2023, pero no hayan tocado durante 2022.

```
SELECT b.nombreBanda, b.genero_musical, b.anio_creacion
FROM Banda b
WHERE (b.codigoB IN (
    SELECT r.codigoB
    FROM Recital r
    WHERE (YEAR(r.fecha) = 2023)
)
AND b.codigoB NOT IN (
    SELECT r.codigoB
    FROM Recital r
    WHERE (YEAR(r.fecha) = 2022)
))
```

(3) Listar el cronograma de recitales del día 04/12/2023. Se deberá listar nombre de la banda que ejecutará el recital, fecha, hora, y el nombre y ubicación del escenario correspondiente.

```
SELECT DISTINCT b.nombreBanda, r.fecha, r.hora, e.nombre_escenario, e.ubicacion
FROM Recital r
```

INNER JOIN Escenario e ON (r.nroEscenario = e.nroEscenario)
INNER JOIN Banda b ON (r.codigoB = b.codigoB)
WHERE (r.fecha = '2023-12-04')

(4) Listar DNI, nombre, apellido, email de integrantes que hayan tocado en el escenario con nombre 'Gustavo Cerati' y en el escenario con nombre 'Carlos Gardel'.

```
SELECT i.DNI, i.nombre, i.apellido, i.email
FROM Recital r
INNER JOIN Escenario e ON (r.nroEscenario = e.nroEscenario)
INNER JOIN Banda b ON (r.codigoB = b.codigoB)
INNER JOIN Integrante i ON (b.codigoB = i.codigoB)
WHERE (e.nombre_escenario = 'Gustavo Cerati')
INTERSECT
SELECT i.DNI, i.nombre, i.apellido, i.email
FROM Recital r
INNER JOIN Escenario e ON (r.nroEscenario = e.nroEscenario)
INNER JOIN Banda b ON (r.codigoB = b.codigoB)
INNER JOIN Integrante i ON (b.codigoB = i.codigoB)
WHERE (e.nombre_escenario = 'Carlos Gardel')
```

(5) Reportar nombre, género musical y año de creación de bandas que tengan más de 5 integrantes.

```
SELECT b.nombreBanda, b.genero_musical, b.anio_creacion
FROM Banda b
INNER JOIN Integrante i ON (b.codigoB = i.codigoB)
GROUP BY b.codigoB, b.nombreBanda, b.genero_musical, b.anio_creacion
HAVING (COUNT(*) > 5)
```

(6) Listar nombre de escenario, ubicación y descripción de escenarios que sólo tuvieron recitales con el género musical rock and roll. Ordenar por nombre de escenario.

```
SELECT e.nombre_escenario, e.ubicacion, e.descripcion
FROM Escenario e
WHERE (e.nroEscenario IN (
    SELECT r.nroEscenario
    FROM Recital r
    INNER JOIN Banda b ON (r.codigoB = b.codigoB)
    WHERE (b.genero_musical IN ('Rock', 'Rock Alternativo', 'Rock Nacional'))
)
AND e.nroEscenario NOT IN (
    SELECT r.nroEscenario
    FROM Recital r
```

```
INNER JOIN Banda b ON (r.codigoB = b.codigoB)
WHERE (b.genero_musical NOT IN ('Rock', 'Rock Alternativo', 'Rock
Nacional'))
)
)
ORDER BY e.nombre_escenario
```

(*) No hay ningún género musical que se llame 'rock and rol', por lo cual se usa 'Rock', 'Rock Alternativo' y 'Rock Nacional'.

(7) Listar nombre, género musical y año de creación de bandas que hayan realizado recitales en escenarios cubiertos durante 2023. // cubierto es true, false según corresponda.

```
SELECT DISTINCT b.nombreBanda, b.genero_musical, b.anio_creacion
FROM Recital r
INNER JOIN Banda b ON (r.codigoB = b.codigoB)
INNER JOIN Escenario e ON (r.nroEscenario = e.nroEscenario)
WHERE (YEAR(r.fecha) = 2023 AND e.cubierto = true)
```

(8) Reportar, para cada escenario, nombre del escenario y cantidad de recitales durante 2024.

```
SELECT e.nombre_escenario, COUNT(r.nroEscenario) AS cantRecitales
FROM Recital r
RIGHT JOIN Escenario e ON (r.nroEscenario = e.nroEscenario)
WHERE (YEAR(r.fecha) = 2024)
GROUP BY e.nroEscenario, e.nombre_escenario
```

(9) Modificar el nombre de la banda 'Mempis la Blusera' a 'Memphis la Blusera'.

```
UPDATE Banda
SET nombreBanda = 'Memphis la Blusera'
WHERE (nombreBanda = 'Mempis la Blusera')
```

Ejercicio 4.

Persona= (DNI, Apellido, Nombre, Fecha_Nacimiento, Estado_Civil, Genero).

Alumno= (DNI(Fk), Legajo, Anio_Ingreso).

Profesor= (DNI(Fk), Matricula, Nro_Expediente).

Titulo= (Cod_Titulo, Nombre, Descripcion).

Titulo_Profesor= (Cod_Titulo(Fk), DNI(Fk), Fecha).

Curso= (Cod_Curso, Nombre, Descripcion, Fecha_Creacion, Duracion).

Alumno_Curso= (DNI(Fk), Cod_Curso(Fk), Anio, Desempenio, Calificacion).

Profesor_Curso= (DNI(Fk), Cod_Curso(Fk), Fecha, Desde, Fecha_Hasta?).

- (1) Listar DNI, legajo y apellido y nombre de todos los alumnos que tengan año de ingreso inferior a 2014.

```
SELECT a.DNI, a.Legajo, p.Apellido, p.Nombre
FROM Alumno a
INNER JOIN Persona p ON (a.DNI = p.DNI)
WHERE (a.Anio_Ingreso < 2014)
```

- (2) Listar DNI, matrícula, apellido y nombre de los profesores que dictan cursos que tengan más de 100 horas de duración. Ordenar por DNI.

```
SELECT DISTINCT p.DNI, p.Matricula, per.Apellido, per.Nombre
FROM Profesor_Curso pc
INNER JOIN Profesor p ON (pc.DNI = p.DNI)
INNER JOIN Persona per ON (pc.DNI = per.DNI)
INNER JOIN Curso c ON (pc.Cod_Curso = c.Cod_Curso)
WHERE (c.Duracion > 100)
ORDER BY p.DNI
```

- (3) Listar DNI, Apellido, Nombre, Género y Fecha de nacimiento de los alumnos inscriptos al curso con nombre 'Diseño de Bases de Datos' en 2023.

```
SELECT per.DNI, per.Apellido, per.Nombre, per.Genero, per.Fecha_Nacimiento
FROM Persona per
WHERE (per.DNI IN (
    SELECT ac.DNI
    FROM Alumno_Curso ac
    INNER JOIN Curso c ON (ac.Cod_Curso = c.Cod_Curso)
    WHERE (ac.anio = 2023 AND c.Nombre = 'Diseño de Bases de Datos')
))
```

(4) Listar DNI, Apellido, Nombre y Calificación de aquellos alumnos que obtuvieron una calificación superior a 8 en algún curso que dicta el profesor ‘Juan García’. Dicho listado deberá estar ordenado por Apellido y nombre.

```
SELECT perA.DNI, perA.Apellido, perA.Nombre, ac.Calificacion
FROM Alumno_Curso ac
INNER JOIN Persona perA ON (ac.DNI = perA.DNI)
INNER JOIN Profesor_Curso pc ON (ac.Cod_Curso = pc.Cod_Curso)
INNER JOIN Persona perP ON (pc.DNI = perP.DNI)
WHERE (ac.Calificacion > 8 AND perP.Nombre = 'Juan' AND perP.Apellido = 'Garcia')
ORDER BY perA.Apellido, perA.Nombre
```

(5) Listar DNI, Apellido, Nombre y Matrícula de aquellos profesores que posean más de 3 títulos. Dicho listado deberá estar ordenado por Apellido y Nombre.

```
SELECT p.DNI, per.Apellido, per.Nombre, p.Matricula
FROM Profesor p
INNER JOIN Persona per ON (p.DNI = per.DNI)
INNER JOIN Titulo_Profesor tp ON (p.DNI = tp.DNI)
GROUP BY p.DNI, per.Apellido, per.Nombre, p.Matricula
HAVING (COUNT(*) > 3)
ORDER BY per.Apellido, per.Nombre
```

(6) Listar DNI, Apellido, Nombre, Cantidad de horas y Promedio de horas que dicta cada profesor. La cantidad de horas se calcula como la suma de la duración de todos los cursos que dicta.

```
SELECT p.DNI, per.Apellido, per.Nombre, SUM(c.Duracion), AVG(c.Duracion)
FROM Profesor p
INNER JOIN Persona per ON (p.DNI = per.DNI)
LEFT JOIN Profesor_Curso pc ON (p.DNI = pc.DNI)
LEFT JOIN Curso c ON (pc.Cod_Curso = c.Cod_Curso)
GROUP BY p.DNI, per.Apellido, per.Nombre
```

(7) Listar Nombre y Descripción del curso que posea más alumnos inscriptos y del que posea menos alumnos inscriptos durante 2024.

```
SELECT c.Nombre, c.Descripcion
FROM Curso c
WHERE (c.Cod_Curso IN (
    SELECT ac.Cod_Curso
    FROM Alumno_Curso ac
    WHERE (ac.anio = 2024)
```

```
GROUP BY ac.Cod_Curso
HAVING (COUNT(*) = (
    SELECT MAX(cant)
    FROM (
        SELECT COUNT(*) AS cant
        FROM Alumno_Curso
        WHERE (Anio = 2024)
        GROUP BY Cod_Curso
    ) AS Max
))
OR COUNT(*) = (
    SELECT MIN(cant)
    FROM (
        SELECT COUNT(*) AS cant
        FROM Alumno_Curso
        WHERE (Anio = 2024)
        GROUP BY Cod_Curso
    ) AS Min
)
)
))
```

(8) Listar el DNI, Apellido, Nombre y Legajo de alumnos que realizaron cursos con nombre conteniendo el string 'BD' durante 2022, pero no realizaron ningún curso durante 2023.

```
SELECT a.DNI, per.Apellido, per.Nombre, a.Legajo
FROM Alumno_Curso ac
INNER JOIN Alumno a ON (ac.DNI = a.DNI)
INNER JOIN Persona per ON (ac.DNI = per.DNI)
INNER JOIN Curso c ON (ac.Cod_Curso = c.Cod_Curso)
WHERE (ac.Anio = 2022 AND c.Nombre LIKE '%BD%')
EXCEPT
SELECT a.DNI, per.Apellido, per.Nombre, a.Legajo
FROM Alumno_Curso ac
INNER JOIN Alumno a ON (ac.DNI = a.DNI)
INNER JOIN Persona per ON (ac.DNI = per.DNI)
WHERE (ac.Anio = 2023)
```

(9) Agregar un profesor con los datos que se prefiera y agregarle el título con código 25.

```
INSERT INTO Persona (DNI, Apellido, Nombre, Fecha_Nacimiento, Estado_Civil, Genero)
VALUES (37102205, 'Menduiña', 'Juan', '1992-09-08', 'Soltero', 'M')
```

```
INSERT INTO Profesor (DNI, Matricula, Nro_Expediente)
```

VALUES (37102205, 'MAT 1000', 10000)

INSERT INTO Titulo_Profesor (Cod_Titulo, DNI, Fecha)
VALUES (25, 37102205, '2025-01-01')

(10) *Modificar el estado civil del alumno cuyo legajo es '2020/09'; el nuevo estado civil es divorciado.*

```
UPDATE Persona
SET Estado_Civil = 'Divorciado'
WHERE (DNI IN (SELECT DNI FROM Alumno WHERE (Legajo = '2020/09')))
```

(11) *Dar de baja el alumno con DNI 30.568.989. Realizar todas las bajas necesarias para no dejar el conjunto de relaciones en un estado inconsistente.*

```
DELETE FROM Alumno_Curso WHERE (DNI = 30568989)
DELETE FROM Alumno WHERE (DNI = 30568989)
DELETE FROM Persona WHERE (DNI = 30568989)
```

Ejercicio 5.

Agencia = (razon_social, direccion, telef, email).

Ciudad = (codigo_postal, nombreCiudad, anioCreacion).

Cliente = (dni, nombre, apellido, telefono, direccion).

Viaje = (fecha, hora, dni(Fk), cpOrigen(Fk), cpDestino(Fk), razon_social(Fk), descripcion). // cpOrigen y cpDestino corresponden a las ciudades origen y destino del viaje, respectivamente.

(1) Listar razón social, dirección y teléfono de agencias que realizaron viajes desde la ciudad de 'La Plata' (ciudad origen) y que el cliente tenga apellido 'Roma'. Ordenar por razón social y, luego, por teléfono.

```
SELECT DISTINCT a.razon_social, a.direccion, a.telef
FROM Viaje v
INNER JOIN Cliente c ON (v.dni = c.dni)
INNER JOIN Ciudad cOrigen ON (v.cpOrigen = cOrigen.codigo_postal)
INNER JOIN Agencia a ON (v.razon_social = a.razon_social)
WHERE (c.apellido = 'Roma' AND cOrigen.nombreCiudad = 'La Plata')
ORDER BY a.razon_social, a.telef
```

(2) Listar fecha, hora, datos personales del cliente, nombres de ciudades origen y destino de viajes realizados en enero de 2019 donde la descripción del viaje contenga el String 'demorado'.

```
SELECT v.fecha, v.hora, c.dni, c.nombre, c.apellido, c.telefono, c.direccion,
cOrigen.nombreCiudad, cDestino.nombreCiudad
FROM Viaje v
INNER JOIN Cliente c ON (v.dni = c.dni)
INNER JOIN Ciudad cOrigen ON (v.cpOrigen = cOrigen.codigo_postal)
INNER JOIN Ciudad cDestino ON (v.cpDestino = cDestino.codigo_postal)
WHERE (YEAR(v.fecha) = 2019 AND v.descripcion LIKE '%demorado%')
```

(3) Reportar información de agencias que realizaron viajes durante 2019 o que tengan dirección de mail que termine con '@jmail.com'.

```
SELECT DISTINCT a.razon_social, a.direccion, a.telef, a.email
FROM Viaje v
INNER JOIN Agencia a ON (v.razon_social = a.razon_social)
WHERE (YEAR(v.fecha) = 2019 OR a.email LIKE '%@jmail.com')
```

(4) Listar datos personales de clientes que viajaron sólo con destino a la ciudad de 'Coronel Brandsen'.

```
SELECT c.dni, c.nombre, c.apellido, c.telefono, c.direccion
FROM Viaje v
INNER JOIN Cliente c ON (v.dni = c.dni)
INNER JOIN Ciudad cDestino ON (v.cpDestino = cDestino.codigo_postal)
WHERE (cDestino.nombreCiudad = 'Coronel Brandsen')
EXCEPT
SELECT c.dni, c.nombre, c.apellido, c.telefono, c.direccion
FROM Viaje v
INNER JOIN Cliente c ON (v.dni = c.dni)
INNER JOIN Ciudad cDestino ON (v.cpDestino = cDestino.codigo_postal)
WHERE (cDestino.nombreCiudad <> 'Coronel Brandsen')
```

(5) Informar cantidad de viajes de la agencia con razón social 'TAXI Y' realizados a Villa Elisa'.

```
SELECT COUNT(*) AS cantViajes
FROM Viaje v
INNER JOIN Ciudad cDestino ON (v.cpDestino = cDestino.codigo_postal)
INNER JOIN Agencia a ON (v.razon_social = a.razon_social)
WHERE (cDestino.nombreCiudad = 'Villa Elisa' AND a.razon_social = 'TAXI Y')
```

(6) Listar nombre, apellido, dirección y teléfono de clientes que viajaron con todas las agencias.

```
SELECT c.nombre, c.apellido, c.direccion, c.telefono
FROM Cliente c
WHERE (NOT EXISTS (
    SELECT *
    FROM Agencia a
    WHERE (NOT EXISTS (
        SELECT *
        FROM Viaje v
        WHERE (a.razon_social = v.razon_social AND c.dni = v.dni)
    )))
))
```

```
SELECT c.nombre, c.apellido, c.direccion, c.telefono
FROM Viaje v
INNER JOIN Cliente c ON (v.dni = c.dni)
INNER JOIN Agencia a ON (v.razon_social = a.razon_social)
GROUP BY c.dni, c.nombre, c.apellido, c.direccion, c.telefono
HAVING (COUNT(DISTINCT a.razon_social) = (
    SELECT COUNT(*)
    FROM Agencia
```

))

(7) Modificar el cliente con DNI 38.495.444 actualizando el teléfono a '221-4400897'.

```
UPDATE Cliente
SET telefono = '221-4400897'
WHERE (dni = 38495444)
```

(8) Listar razón social, dirección y teléfono de la/s agencias que tengan mayor cantidad de viajes realizados.

```
SELECT a.razon_social, a.direccion, a.telef
FROM Viaje v
INNER JOIN Agencia a ON (v.razon_social = a.razon_social)
GROUP BY a.razon_social, a.direccion, a.telef
HAVING (COUNT(*) >= ALL (
    SELECT COUNT(*)
    FROM Viaje v
    GROUP BY v.razon_social
))
```

(9) Reportar nombre, apellido, dirección y teléfono de clientes con, al menos, 5 viajes.

```
SELECT c.nombre, c.apellido, c.direccion, c.telefono
FROM Viaje v
INNER JOIN Cliente c ON (v.dni = c.dni)
GROUP BY c.dni, c.nombre, c.apellido, c.telefono, c.direccion
HAVING (COUNT(*) >= 5)
```

(10) Borrar al cliente con DNI 40.325.692.

```
DELETE FROM Viaje WHERE (dni = 40325692)
DELETE FROM Cliente WHERE (dni = 40325692)
```

Ejercicio 6.

Tecnico= (codTec, nombre, especialidad). // técnicos.

Repuesto= (codRep, nombre, stock, precio). // repuestos.

RepuestoReparacion= (nroReparac(Fk), codRep(Fk), cantidad, precio). // repuestos utilizados en reparaciones.

Reparacion= (nroReparac, codTec(Fk), precio_total, fecha). // reparaciones realizadas.

- (1) Listar los repuestos, informando nombre, stock y precio. Ordenar el resultado por precio.

```
SELECT nombre, stock, precio
FROM Repuesto
ORDER BY Precio
```

- (2) Listar nombre, stock y precio de repuestos que se usaron en reparaciones durante 2023 y que no se usaron en reparaciones del técnico 'José Gonzalez'.

```
SELECT rep.nombre, rep.stock, rep.precio
FROM Repuesto rep
WHERE (rep.codRep IN (
    SELECT rr.codRep
    FROM RepuestoReparacion rr
    INNER JOIN Reparacion r ON (rr.nroReparac = r.nroReparac)
    WHERE (YEAR(r.fecha) = 2023)
)
AND rep.codRep NOT IN (
    SELECT rr.codRep
    FROM RepuestoReparacion rr
    INNER JOIN Reparacion r ON (rr.nroReparac = r.nroReparac)
    INNER JOIN Tecnico t ON (r.codTec = t.codTec)
    WHERE (t.nombre = 'José Gonzalez')
))
```

- (3) Listar nombre y especialidad de técnicos que no participaron en ninguna reparación. Ordenar por nombre ascendente.

```
SELECT t.nombre, t.especialidad
FROM Tecnico t
WHERE (t.codTec NOT IN (
    SELECT r.codTec
    FROM Reparacion r
```

))
ORDER BY t.nombre

(4) Listar nombre y especialidad de los técnicos que sólo participaron en reparaciones durante 2022.

```
SELECT t.nombre, t.especialidad
FROM Tecnico t
WHERE (t.codTec IN (
    SELECT r.codTec
    FROM Reparacion r
    WHERE (YEAR(r.fecha) = 2022)
))
AND t.codTec NOT IN (
    SELECT r.codTec
    FROM Reparacion r
    WHERE (YEAR(r.fecha) <> 2022)
)
)
```

(5) Listar, para cada repuesto, nombre, stock y cantidad de técnicos distintos que lo utilizaron. Si un repuesto no participó en alguna reparación, igual debe aparecer en dicho listado.

```
SELECT rep.nombre, rep.stock, COUNT(DISTINCT r.codTec) AS cantUsados
FROM RepuestoReparacion rr
INNER JOIN Reparacion r ON (rr.nroReparac = r.nroReparac)
RIGHT JOIN Repuesto rep ON (rr.codRep = rep.codRep)
GROUP BY rep.codRep, rep.nombre, rep.stock
```

(6) Listar nombre y especialidad del técnico con mayor cantidad de reparaciones realizadas y el técnico con menor cantidad de reparaciones.

```
SELECT t.nombre, t.especialidad
FROM Reparacion r
INNER JOIN Tecnico t ON (r.codTec = t.codTec)
GROUP BY t.codTec, t.nombre, t.especialidad
HAVING (COUNT(*) >= ALL (
    SELECT COUNT(*)
    FROM Reparacion r
    GROUP BY r.codTec
))
OR COUNT(*) <= ALL (
    SELECT COUNT(*)
```

```
FROM Reparacion r
GROUP BY r.codTec
)
)
```

(7) Listar nombre, stock y precio de todos los repuestos con stock mayor a 0 y que dicho repuesto no haya estado en reparaciones con un precio total superior a \$10.000.

```
SELECT rep.nombre, rep.stock, rep.precio
FROM Repuesto rep
WHERE (rep.stock > 0 AND rep.codRep NOT IN (
    SELECT rr.codRep
    FROM RepuestoReparacion rr
    INNER JOIN Reparacion r ON (rr.nroReparac = r.nroReparac)
    WHERE (r.precio_total > 10000)
))
```

(8) Proyectar número, fecha y precio total de aquellas reparaciones donde se utilizó algún repuesto con precio en el momento de la reparación mayor a \$10.000 y menor a \$15.000.

```
SELECT r.nroReparac, r.fecha, r.precio_total
FROM RepuestoReparacion rr
INNER JOIN Reparacion r ON (rr.nroReparac = r.nroReparac)
WHERE (rr.precio BETWEEN 10000 AND 15000)
```

(9) Listar nombre, stock y precio de repuestos que hayan sido utilizados por todos los técnicos.

```
SELECT rep.nombre, rep.stock, rep.precio
FROM RepuestoReparacion rr
INNER JOIN Reparacion r ON (rr.nroReparac = r.nroReparac)
INNER JOIN Repuesto rep ON (rr.codRep = rep.codRep)
GROUP BY rep.codRep, rep.nombre, rep.stock, rep.precio
HAVING (COUNT(DISTINCT r.codTec) =
    SELECT COUNT(*)
    FROM Tecnico
))
```

(10) Listar fecha, técnico y precio total de aquellas reparaciones que necesitaron, al menos, 4 repuestos distintos.

```
SELECT r.fecha, t.nombre, r.precio_total
FROM RepuestoReparacion rr
INNER JOIN Reparacion r ON (rr.nroReparac = r.nroReparac)
INNER JOIN Tecnico t ON (r.codTec = t.codTec)
GROUP BY r.nroReparac, r.fecha, t.nombre, r.precio_total
HAVING (COUNT(*) >= 4)
```

Ejercicio 7.

Club= (codigoClub, nombre, anioFundacion, codigoCiudad(Fk)).

Ciudad= (codigoCiudad, nombre).

Estadio= (codigoEstadio, codigoClub(Fk), nombre, direccion).

Jugador= (DNI, nombre, apellido, edad, codigoCiudad(Fk)).

ClubJugador= (codigoClub(Fk), DNI(Fk), desde, hasta).

(1) Reportar nombre y año de fundación de aquellos clubes de la ciudad de La Plata que no poseen estadio.

```
SELECT c.nombre, c.anioFundacion
FROM Club c
INNER JOIN Ciudad ciu ON (c.codigoCiudad = ciu.codigoCiudad)
WHERE (ciu.nombre = 'La Plata' AND c.codigoClub NOT IN (
    SELECT e.codigoClub
    FROM Estadio e
))
```

(2) Listar nombre de los clubes que no hayan tenido ni tengan jugadores de la ciudad de Berisso.

```
SELECT c.nombre
FROM Club c
WHERE (c.codigoClub NOT IN (
    SELECT DISTINCT cj.codigoClub
    FROM Jugador j
    INNER JOIN ClubJugador cj ON (j.DNI = cj.DNI)
    INNER JOIN Ciudad ciu ON (j.codigoCiudad = ciu.codigoCiudad)
    WHERE (ciu.nombre = 'Berisso')
))
```

(3) Mostrar DNI, nombre y apellido de aquellos jugadores que jugaron o juegan en el club 'Gimnasia y Esgrima La Plata'.

```
SELECT j.DNI, j.nombre, j.apellido
FROM Jugador j
INNER JOIN ClubJugador cj ON (j.DNI = cj.DNI)
INNER JOIN Club c ON (cj.codigoClub = c.codigoClub)
WHERE (c.nombre = 'Gimnasia y Esgrima La Plata')
```

(4) Mostrar DNI, nombre y apellido de aquellos jugadores que tengan más de 29 años y hayan jugado o juegan en algún club de la ciudad de Córdoba.

```
SELECT DISTINCT j.DNI, j.nombre, j.apellido
FROM Jugador j
INNER JOIN ClubJugador cj ON (j.DNI = cj.DNI)
INNER JOIN Club c ON (cj.codigoClub = c.codigoClub)
INNER JOIN Ciudad ciU ON (c.codigoCiudad = ciu.codigoCiudad)
WHERE (j.edad > 29 AND ciu.nombre = 'Córdoba')
```

(5) Mostrar, para cada club, nombre de club y la edad promedio de los jugadores que juegan, actualmente, en cada uno.

```
SELECT c.nombre, AVG(j.edad) AS edadProm
FROM Jugador j
INNER JOIN ClubJugador cj ON (j.DNI = cj.DNI)
INNER JOIN Club c ON (cj.codigoClub = c.codigoClub)
WHERE (cj.hasta IS NULL)
GROUP BY c.codigoClub, c.nombre
```

(6) Listar, para cada jugador, nombre, apellido, edad y cantidad de clubes diferentes en los que jugó (incluido el actual).

```
SELECT j.nombre, j.apellido, j.edad, COUNT(DISTINCT cj.codigoClub) AS cantClubes
FROM Jugador j
INNER JOIN ClubJugador cj ON (j.DNI = cj.DNI)
GROUP BY j.DNI, j.nombre, j.apellido, j.edad
```

(7) Mostrar el nombre de los clubes que nunca hayan tenido jugadores de la ciudad de Mar del Plata.

```
SELECT c.nombre
FROM Club c
WHERE (c.codigoClub NOT IN (
    SELECT DISTINCT cj.codigoClub
    FROM Jugador j
    INNER JOIN ClubJugador cj ON (j.DNI = cj.DNI)
    INNER JOIN Ciudad ciu ON (j.codigoCiudad = ciu.codigoCiudad)
    WHERE (ciu.nombre = 'Mar del Plata')
))
```

(8) Reportar el nombre y apellido de aquellos jugadores que hayan jugado en todos los clubes de la ciudad de Córdoba.

```
SELECT j.nombre, j.apellido
FROM Jugador j
INNER JOIN ClubJugador cj ON (j.DNI = cj.DNI)
INNER JOIN Club c ON (cj.codigoClub = c.codigoClub)
INNER JOIN Ciudad ciu ON (c.codigoCiudad = ciu.codigoCiudad)
WHERE (ciu.nombre = 'Córdoba')
GROUP BY j.DNI, j.nombre, j.apellido
HAVING (COUNT(DISTINCT c.codigoClub) = (
    SELECT COUNT(*)
    FROM Club c
    INNER JOIN Ciudad ciu ON (c.codigoCiudad = ciu.codigoCiudad)
    WHERE (ciu.nombre = 'Córdoba'))
))
```

(9) Agregar el club “Estrella de Berisso”, con código 1234, que se fundó en 1921 y que pertenece a la ciudad de Berisso. Puede asumirse que el códigoClub 1234 no existe en la tabla Club.

```
INSERT INTO Club (codigoClub, nombre, anioFundacion, codigoCiudad)
VALUES (1234, 'Estrella de Berisso', 1921, (
    SELECT codigoCiudad
    FROM Ciudad
    WHERE (nombre = 'Berisso'))
))
```

Ejercicio 8.

Equipo= (codigoE, nombreE, descripcionE).

Integrante= (DNI, nombre, apellido, ciudad, email, telefono, codigoE(Fk)).

Laguna= (nroLaguna, nombreL, ubicacion, extension, descripcion).

TorneoPesca= (codTorneo, fecha, hora, nroLaguna(Fk), descripcion).

Inscripcion= (codTorneo(Fk), codigoE(Fk), asistio, gano). // asistio y gano son true/false según corresponda.

(1) Listar DNI, nombre, apellido y email de integrantes que sean de la ciudad 'La Plata' y estén inscriptos en torneos disputados en 2023.

```
SELECT DISTINCT i.DNI, i.nombre, i.apellido, i.email
FROM Integrante i
INNER JOIN Inscripcion ins ON (i.codigoE = ins.codigoE)
INNER JOIN TorneoPesca t ON (ins.codTorneo = t.codTorneo)
WHERE (i.ciudad = 'La Plata' AND YEAR(t.fecha) = 2023)
```

(2) Reportar nombre y descripción de equipos que sólo se hayan inscripto en torneos de 2020.

```
SELECT e.nombreE, e.descripcionE
FROM Equipo e
WHERE (e.codigoE IN (
    SELECT ins.codigoE
    FROM Inscripcion ins
    INNER JOIN TorneoPesca t ON (ins.codTorneo = t.codTorneo)
    WHERE (YEAR(t.fecha) = 2020)
)
AND e.codigoE NOT IN (
    SELECT ins.codigoE
    FROM Inscripcion ins
    INNER JOIN TorneoPesca t ON (ins.codTorneo = t.codTorneo)
    WHERE (YEAR(t.fecha) <> 2020)
)
)
```

(3) Listar DNI, nombre, apellido, email y ciudad de integrantes que asistieron a torneos en la laguna con nombre 'La Salada, Coronel Granada' y su equipo no tenga inscripciones a torneos disputados en 2023.

```
SELECT DISTINCT i.DNI, i.nombre, i.apellido, i.email, i.ciudad
FROM Integrante i
INNER JOIN Inscripcion ins ON (i.codigoE = ins.codigoE)
```

```
INNER JOIN TorneoPesca t ON (ins.codTorneo = t.codTorneo)
INNER JOIN Laguna l ON (t.nroLaguna = l.nroLaguna)
WHERE (ins.asistio = TRUE AND l.nombreL = 'La Salada, Coronel Granada' AND
i.codigoE NOT IN (
    SELECT ins.codigoE
    FROM Inscripcion ins
    INNER JOIN TorneoPesca t ON (ins.codTorneo = t.codTorneo)
    WHERE (YEAR(t.fecha) = 2023)
))
```

(4) Reportar nombre y descripción de equipos que tengan, al menos 5, integrantes. Ordenar por nombre.

```
SELECT e.nombreE, e.descripcionE
FROM Integrante i
INNER JOIN Equipo e ON (i.codigoE = e.codigoE)
GROUP BY e.codigoE, e.nombreE, e.descripcionE
HAVING (COUNT(*) >= 5)
ORDER BY e.nombreE
```

(5) Reportar nombre y descripción de equipos que tengan inscripciones en todas las lagunas.

```
SELECT e.nombreE, e.descripcionE
FROM Inscripcion ins
INNER JOIN TorneoPesca t ON (ins.codTorneo = t.codTorneo)
INNER JOIN Equipo e ON (ins.codigoE = e.codigoE)
INNER JOIN Laguna l ON (t.nroLaguna = l.nroLaguna)
GROUP BY e.codigoE, e.nombreE, e.descripcionE
HAVING (COUNT(DISTINCT l.nroLaguna) = (
    SELECT COUNT(*)
    FROM Laguna
))
```

(6) Eliminar el equipo con código 10.000.

```
DELETE FROM Integrante WHERE (codigoE = 10000)
DELETE FROM Inscripcion WHERE (codigoE = 10000)
DELETE FROM Equipo WHERE (codigoE = 10000)
```

(7) Listar nombre, ubicación, extensión y descripción de lagunas que no tuvieron torneos.

```
SELECT l.nombreL, l.ubicacion, l.extension, l.descripcion
FROM Laguna l
WHERE (l.nroLaguna NOT IN (
    SELECT t.nroLaguna
    FROM TorneoPesca t
))
```

(8) Reportar nombre y descripción de equipos que tengan inscripciones a torneos a disputarse durante 2024, pero no tienen inscripciones a torneos de 2023.

```
SELECT e.nombreE, e.descripcionE
FROM Equipo e
WHERE (e.codigoE IN (
    SELECT ins.codigoE
    FROM Inscripcion ins
    INNER JOIN TorneoPesca t ON (ins.codTorneo = t.codTorneo)
    WHERE (YEAR(t.fecha) = 2024)
)
AND e.codigoE NOT IN (
    SELECT ins.codigoE
    FROM Inscripcion ins
    INNER JOIN TorneoPesca t ON (ins.codTorneo = t.codTorneo)
    WHERE (YEAR(t.fecha) = 2023)
)
)
```

(9) Listar DNI, nombre, apellido, ciudad y email de integrantes que ganaron algún torneo que se disputó en la laguna con nombre 'Laguna de Chascomús'.

```
SELECT DISTINCT i.DNI, i.nombre, i.apellido, i.ciudad, i.email
FROM Inscripcion ins
INNER JOIN TorneoPesca t ON (ins.codTorneo = t.codTorneo)
INNER JOIN Equipo e ON (ins.codigoE = e.codigoE)
INNER JOIN Laguna l ON (t.nroLaguna = l.nroLaguna)
INNER JOIN Integrante i ON (e.codigoE = i.codigoE)
WHERE (ins.gano = TRUE AND l.nombreL = 'Laguna de Chascomús')
```

Ejercicio 9.

Proyecto= (codProyecto, nombreP, descripcion, fechaInicioP, fechaFinP?, fechaFinEstimada, DNIResponsable(Fk), equipoBackend(Fk), equipoFrontend(Fk)). // DNIResponsable corresponde a un empleado, equipoBackend y equipoFrontend corresponden a un equipo.

Equipo= (codEquipo, nombreE, descTecnologias, DNILider(Fk)). // DNILider corresponde a un empleado.

Empleado= (DNI, nombre, apellido, telefono, direccion, fechaIngreso).

Empleado_Equipo= (codEquipo(Fk), DNI(Fk), fechaInicio, fechaFin?, descripcionRol).

(1) Listar nombre, descripción, fecha de inicio y fecha de fin de proyectos ya finalizados que no fueron terminados antes de la fecha de fin estimada.

```
SELECT nombreP, descripcion, fechaInicioP, fechaFinP
FROM Proyecto
WHERE (fechaFinP IS NOT NULL AND fechaFinP >= fechaFinEstimada)
```

(2) Listar DNI, nombre, apellido, teléfono, dirección y fecha de ingreso de empleados que no son ni fueron responsables de proyectos. Ordenar por apellido y nombre.

```
SELECT emp.DNI, emp.nombre, emp.apellido, emp.telefono, emp.direccion,
emp.fechaIngreso
FROM Empleado emp
WHERE (emp.DNI NOT IN (
    SELECT DISTINCT p.DNIResponsable
    FROM Proyecto p
))
ORDER BY emp.apellido, emp.nombre
```

(3) Listar DNI, nombre, apellido, teléfono y dirección de líderes de equipo que tenga más de un equipo a cargo.

```
SELECT emp.DNI, emp.nombre, emp.apellido, emp.telefono, emp.direccion
FROM Equipo e
INNER JOIN Empleado emp ON (e.DNILider = emp.DNI)
GROUP BY emp.DNI, emp.nombre, emp.apellido, emp.telefono, emp.direccion
HAVING (COUNT(*) > 1)
```

(4) Listar DNI, nombre, apellido, teléfono y dirección de todos los empleados que trabajan en el proyecto con nombre 'Proyecto X'. No es necesario informar responsable y líderes.

```
SELECT DISTINCT emp.DNI, emp.nombre, emp.apellido, emp.telefono, emp.direccion
FROM Proyecto p
INNER JOIN Empleado_Equipo ee ON (p.equipoBackend = ee.codEquipo OR
p.equipoFrontend = ee.codEquipo)
INNER JOIN Empleado emp ON (ee.DNI = emp.DNI)
WHERE (p.fechaFinP IS NULL AND p.nombrP = 'Proyecto X')
```

(5) Listar nombre de equipo y datos personales de líderes de equipos que no tengan empleados asignados y trabajen con tecnología 'Java'.

```
SELECT e.nombreE, emp.DNI, emp.nombre, emp.apellido, emp.telefono, emp.direccion
FROM Empleado_Equipo ee
RIGHT JOIN Equipo e ON (ee.codEquipo = e.codEquipo)
INNER JOIN Empleado emp ON (e.DNILider = emp.DNI)
WHERE (ee.codEquipo IS NULL AND eqdescTecnologias LIKE '%Java%')
```

(6) Modificar nombre, apellido y dirección del empleado con DNI 40.568.965 con los datos que se deseé.

```
UPDATE Empleado
SET nombre = 'Juan', apellido = 'Menduiña', direccion = '13 Nro. 22'
WHERE (DNI = 40568965)
```

(7) Listar DNI, nombre, apellido, teléfono y dirección de empleados que son responsables de proyectos, pero no han sido líderes de equipo.

```
SELECT emp.DNI, emp.nombre, emp.apellido, emp.telefono, emp.direccion
FROM Empleado emp
WHERE (emp.DNI IN (
    SELECT p.DNIResponsable
    FROM Proyecto p
    WHERE (p.fechaFinP IS NULL)
)
AND emp.DNI NOT IN (
    SELECT e.DNILider
    FROM Equipo e
)
)
```

(8) Listar nombre de equipo y descripción de tecnologías de equipos que hayan sido asignados como equipos frontend y backend.

```
SELECT e.nombreE, e.descTecnologias
FROM Equipo e
WHERE (e.codEquipo IN (
    SELECT p.equipoFrontend
    FROM Proyecto p
)
AND e.codEquipo IN (
    SELECT p.equipoBackend
    FROM Proyecto p
)
)
```

(9) Listar nombre, descripción, fecha de inicio, nombre y apellido de responsables de proyectos que se estiman finalizar durante 2025.

```
SELECT p.nombrP, p.descripcion, p.fechaInicioP, emp.nombre, emp.apellido
FROM Proyecto p
INNER JOIN Empleado emp ON (p.DNIResponsable = emp.DNI)
WHERE (p.fechaFinP IS NULL AND YEAR(p.fechaFinEstimada) = 2025)
```

Ejercicio 10.

Vehiculo= (patente, modelo, marca, peso, km).

Camion= (patente(Fk), largo, max_toneladas, cant_ruedas, tiene_acoplado).

Auto= (patente(Fk), es_electrico, tipo_motor).

Service= (fecha, patente(Fk), km_service, descripcion, monto).

Parte= (cod_parte, nombre, precio_parte).

Service_Parte= ([fecha, patente](Fk), cod_parte(Fk), precio).

- (1) Listar todos los datos de aquellos camiones que tengan entre 4 y 8 ruedas, y que hayan realizado algún service en los últimos 365 días. Ordenar por marca, modelo y patente.

```
SELECT v.patente, v.modelo, v.marca, v.peso, v.km, c.largo, c.max_toneladas,
c.cant_ruedas, c.tiene_acoplado
FROM Service s
INNER JOIN Vehiculo v ON (s.patente = v.patente)
INNER JOIN Camion c ON (s.patente = c.patente)
WHERE (c.cant_ruedas BETWEEN 4 AND 8 AND s.fecha >= CURRENT_DATE -
INTERVAL '365' DAY)
ORDER BY v.marca, v.modelo, v.patente
```

- (2) Listar los autos que hayan realizado el service “cambio de aceite” antes de los 13.000 km o hayan realizado el service “inspección general” que incluya la parte “Filtro de combustible”.

```
SELECT v.patente, v.modelo, v.marca, v.peso, v.km, a.es_electrico, a.tipo_motor
FROM Service_Parte sp
INNER JOIN Service s ON (sp.fecha = s.fecha AND sp.patente = s.patente)
INNER JOIN Parte p ON (sp.cod_parte = p.cod_parte)
INNER JOIN Vehiculo v ON (sp.patente = v.patente)
INNER JOIN Auto a ON (sp.patente = a.patente)
WHERE ((s.descripcion = 'cambio de aceite' AND s.km_service < 13000) OR
(s.descripcion = 'inspección general' AND p.nombre = 'Filtro de combustible'))
```

- (3) Listar nombre y precio de todas las partes que aparezcan en más de 30 services que hayan salido (partes) más de \$4.000.

```
SELECT p.nombre, p.precio_parte
FROM Service_Parte sp
INNER JOIN Service s ON (sp.fecha = s.fecha AND sp.patente = s.patente)
INNER JOIN Parte p ON (sp.cod_parte = p.cod_parte)
WHERE (p.precio_parte > 4000)
```

GROUP BY p.cod_parte, p.nombre, p.precio_parte
HAVING (COUNT(*) > 30)

(4) Dar de baja todos los camiones con más de 250.000 km.

```
DELETE FROM Service_Parte sp
INNER JOIN Service s ON (sp.fecha = s.fecha AND sp.patente = s.patente)
WHERE (s.patente IN (
    SELECT v.patente
    FROM Vehiculo v
    WHERE (v.km > 250000)
```

```
))
```

```
DELETE FROM Service s
WHERE (s.patente IN (
    SELECT v.patente
    FROM Vehiculo v
    WHERE (v.km > 250000)
))
```

```
DELETE FROM Camion c
WHERE (c.patente IN (
    SELECT v.patente
    FROM Vehiculo v
    WHERE (v.km > 250000)
))
```

```
DELETE FROM Vehiculo WHERE (km > 250000)
```

(5) Listar el nombre y precio de aquellas partes que figuren en todos los services realizados en el año actual.

```
SELECT p.nombre, p.precio_parte
FROM Service_Parte sp
INNER JOIN Parte p ON (sp.cod_parte = p.cod_parte)
WHERE (YEAR(sp.fecha) = YEAR(CURRENT_DATE))
GROUP BY p.cod_parte, p.nombre, p.precio_parte
HAVING (COUNT(DISTINCT sp.fecha, sp.patente) = (
    SELECT COUNT(*)
    FROM Service s
    WHERE (YEAR(s.fecha) = YEAR(CURRENT_DATE)))
))
```

(6) Listar todos los autos que sean eléctricos. Mostrar información de patente, modelo, marca y peso.

```
SELECT v.patente, v.modelo, v.marca, v.peso
FROM Vehiculo v
INNER JOIN Auto a ON (v.patente = a.patente)
WHERE (a.es_electrico = 1)
```

(7) *Dar de alta una parte, cuyo nombre sea “Aleron” y precio \$5.000.*

```
INSERT INTO Parte (cod_parte, nombre, precio_parte)
VALUES (100, 'Aleron', 5000)
```

(8) *Dar de baja todos los services que se realizaron al auto con patente ‘AWA564’.*

```
DELETE FROM Service_Parte WHERE (patente = 'AWA564')
DELETE FROM Service WHERE (patente = 'AWA564')
```

(9) *Listar todos los vehículos que hayan tenido services durante el 2024.*

```
SELECT DISTINCT v.patente, v.modelo, v.marca, v.peso, v.km
FROM Service s
INNER JOIN Vehiculo v ON (s.patente = v.patente)
WHERE (YEAR(s.fecha) = 2024)
```

Ejercicio 11.

Box= (nroBox, m2, ubicacion, capacidad, ocupacion). // ocupación es un numérico indicando cantidad de mascotas en el box actualmente, capacidad es una descripción.

Mascota= (codMascota, nombre, edad, raza, peso, telefonoContacto).

Veterinario= (matricula, CUIT, nombYAp, direccion, telefono).

Supervision= (codMascota(Fk), nroBox(Fk), fechaEntra, fechaSale?, matricula(Fk), descripcionEstadia). // fechaSale tiene valor null si la mascota está, actualmente, en el box.

(1) Listar, para cada veterinario, cantidad de supervisiones realizadas con fecha de salida (fechaSale) durante enero de 2024. Indicar matrícula, CUIT, nombre y apellido, dirección, teléfono y cantidad de supervisiones.

```
SELECT      v.matricula,      v.CUIT,      v.nombYAp,      v.direccion,      v.telefono,  
COUNT(s.matricula) AS cantSupervisiones  
FROM Supervision s  
RIGHT JOIN Veterinario v ON (s.matricula = v.matricula)  
WHERE (s.fechaSale IS NOT NULL AND YEAR(s.fechaSale) = 2024 AND  
MONTH(s.fechaSale) = 01)  
GROUP BY v.matricula, v.CUIT, v.nombYAp, v.direccion, v.telefono
```

(2) Listar CUIT, matrícula, nombre, apellido, dirección y teléfono de veterinarios que no tengan mascotas bajo supervisión actualmente.

```
SELECT v.CUIT, v.matricula, v.nombYAp, v.direccion, v.telefono  
FROM Veterinario v  
WHERE (v.matricula NOT IN (  
    SELECT s.matricula  
    FROM Supervision s  
    WHERE (s.fechaSale IS NULL)  
))
```

(3) Listar nombre, edad, raza, peso y teléfono de contacto de mascotas que fueron atendidas por el veterinario 'Oscar Lopez'. Ordenar por nombre y raza de manera ascendente.

```
SELECT m.nombre, m.edad, m.raza, m.peso, m.telefonoContacto  
FROM Supervision s  
INNER JOIN Mascota m ON (s.codMascota = m.codMascota)  
INNER JOIN Veterinario v ON (s.matricula = v.matricula)  
WHERE (s.fechaSale IS NOT NULL AND v.nombYAp = 'Oscar Lopez')  
ORDER BY m.nombre, m.raza
```

- (4) Modificar nombre y apellido al veterinario con matrícula 'MP 10000'; deberá llamarse 'Pablo Lopez'.

```
UPDATE Veterinario
SET nombYAp = 'Pablo Lopez'
WHERE (matricula = 'MP 10000')
```

- (5) Listar nombre, edad, raza y peso de mascotas que tengan supervisiones con el veterinario con matrícula 'MP 1000' y con el veterinario con matrícula 'MN 4545'.

```
SELECT m.nombre, m.edad, m.raza, m.peso
FROM Mascota m
WHERE (EXISTS (
    SELECT 1
    FROM Supervision s
    WHERE (m.codMascota = s.codMascota AND s.matricula = 'MP 1000')
))
AND EXISTS (
    SELECT 1
    FROM Supervision s
    WHERE (m.codMascota = s.codMascota AND s.matricula = 'MP 4545')
)
)
```

- (6) Listar número de box, m2, ubicación, capacidad y nombre de mascota para supervisiones con fecha de entrada durante 2024.

```
SELECT b.nroBox, b.m2, b.ubicacion, b.capacidad, m.nombre
FROM Supervision s
INNER JOIN Mascota m ON (s.codMascota = m.codMascota)
INNER JOIN Box b ON (s.nroBox = b.nroBox)
WHERE (YEAR(s.fechaEntra) = 2024)
```

Ejercicio 12.

Barberia = (codBarberia, razon_social, direccion, telefono).

Cliente = (nroCliente, DNI, nombYAp, direccionC, fechaNacimiento, celular).

Barbero = (codEmpleado, DNIB, nombYApB, direccionB, telefonoContacto, mail).

Atencion = (codEmpleado(Fk), fecha, hora, codBarberia(Fk), nroCliente(Fk), descTratamiento, valor).

- (1)** Listar DNI, nombre y apellido, dirección, fecha de nacimiento y celular de clientes que no tengan atenciones durante 2024.

```
SELECT c.DNI, c.nombYAp, c.direccionC, c.fechaNacimiento, c.celular
FROM Cliente c
WHERE (c.nroCliente NOT IN (
    SELECT a.nroCliente
    FROM Atencion a
    WHERE (YEAR(a.fecha) = 2024)
))
```

- (2)** Listar, para cada barbero, cantidad de atenciones que realizaron durante 2023.
Listar DNI, nombre y apellido, dirección, teléfono de contacto, mail y cantidad de atenciones.

```
SELECT b.DNIB, b.nombYApB, b.direccionB, b.telefonoContacto, b.mail,
COUNT(a.codEmpleado) AS cantAtenciones
FROM Atencion a
RIGHT JOIN Barbero b ON (a.codEmpleado = b.codEmpleado)
WHERE (YEAR(a.fecha) = 2023)
GROUP BY b.DNIB, b.nombYApB, b.direccionB, b.telefonoContacto, b.mail
```

- (3)** Listar razón social, dirección y teléfono de barberías que tengan atenciones para el cliente con DNI 22.283.566. Ordenar por razón social y dirección ascendente.

```
SELECT DISTINCT bar.razon_social, bar.direccion, bar.telefono
FROM Atencion a
INNER JOIN Barberia bar ON (a.codBarberia = bar.codBarberia)
INNER JOIN Cliente c ON (a.nroCliente = c.nroCliente)
WHERE (c.DNI = 22283566)
ORDER BY bar.razon_social, bar.direccion
```

- (4)** Listar DNI, nombre y apellido, dirección, teléfono de contacto y mail de barberos que tengan atenciones con valor superior a \$5.000.

```
SELECT DISTINCT b.DNIB, b.nombYApB, b.direccionB, b.telefonoContacto, b.mail
FROM Atencion a
INNER JOIN Barbero b ON (a.codEmpleado = b.codEmpleado)
WHERE (a.valor > 5000)
```

(5) Listar DNI, nombYAp, direccionC, fechaNacimiento y celular de clientes que tengan atenciones en la barbería con razón social ‘Corta barba’ y también se hayan atendido en la barbería con razón social ‘Barberia Barbara’.

```
SELECT c.DNI, c.nombYAp, c.direccionC, c.fechaNacimiento, c.celular
FROM Atencion a
INNER JOIN Barberia b ON (a.codBarberia = b.codBarberia)
INNER JOIN Cliente c ON (a.nroCliente = c.nroCliente)
WHERE (b.razon_social = 'Corta barba')
INTERSECT
SELECT c.DNI, c.nombYAp, c.direccionC, c.fechaNacimiento, c.celular
FROM Atencion a
INNER JOIN Barberia b ON (a.codBarberia = b.codBarberia)
INNER JOIN Cliente c ON (a.nroCliente = c.nroCliente)
WHERE (b.razon_social = 'Barberia Barbara')
```

(6) Eliminar el cliente con DNI 22.222.222.

```
DELETE FROM Atencion a
WHERE (a.nroCliente IN (
    SELECT c.nroCliente
    FROM Cliente c
    WHERE (c.DNI = 22222222)
))
DELETE FROM Cliente WHERE (DNI = 22222222)
```

Ejercicio 13.

Club = (IdClub, nombreClub, ciudad).

Complejo = (IdComplejo, nombreComplejo, IdClub(Fk)).

Cancha = (IdCancha, nombreCancha, IdComplejo(Fk)).

Entrenador = (IdEntrenador, nombreEntrenador, fechaNacimiento, direccion).

Entrenamiento = (IdEntrenamiento, fecha, IdEntrenador(Fk), IdCancha(Fk)).

- (1)** Listar nombre, fecha de nacimiento y dirección de entrenadores que hayan tenido entrenamientos durante 2023.

```
SELECT DISTINCT ent.nombreEntrenador, ent.fechaNacimiento, ent.direccion
FROM Entrenamiento e
INNER JOIN Entrenador ent ON (e.IdEntrenador = ent.IdEntrenador)
WHERE (YEAR(e.fecha) = 2023)
```

- (2)** Listar, para cada cancha del complejo “Complejo 1”, la cantidad de entrenamientos que se realizaron durante el 2022. Informar nombre de la cancha y cantidad de entrenamientos.

```
SELECT c.idCancha, c.nombreCancha, COUNT(e.IdCancha) AS cantEntrenamientos
FROM Cancha c
INNER JOIN Complejo com ON (c.IdComplejo = com.IdComplejo)
LEFT JOIN Entrenamiento e ON (c.IdCancha = e.IdCancha)
WHERE (com.nombreComplejo = 'Complejo 1' AND YEAR(e.fecha) = 2022)
GROUP BY c.idCancha, c.nombreCancha
```

- (3)** Listar los complejos donde haya realizado entrenamientos el entrenador ‘Jorge Gonzalez’. Informar nombre de complejo. Ordenar el resultado de manera ascendente.

```
SELECT DISTINCT com.nombreComplejo
FROM Entrenamiento e
INNER JOIN Entrenador ent ON (e.IdEntrenador = ent.IdEntrenador)
INNER JOIN Cancha c ON (e.IdCancha = c.IdCancha)
INNER JOIN Complejo com ON (c.IdComplejo = com.IdComplejo)
WHERE (ent.nombreEntrenador = 'Jorge Gonzalez')
ORDER BY com.nombreComplejo
```

- (4)** Listar nombre, fecha de nacimiento y dirección de entrenadores que hayan entrenado en los clubes con nombre ‘Everton’ y ‘Estrella de Berisso’.

```
SELECT ent.nombreEntrenador, ent.fechaNacimiento, ent.direccion
```

```
FROM Entrenamiento e
INNER JOIN Entrenador ent ON (e.IdEntrenador = ent.IdEntrenador)
INNER JOIN Cancha c ON (e.IdCancha = c.IdCancha)
INNER JOIN Complejo com ON (c.IdComplejo = com.IdComplejo)
INNER JOIN Club clu ON (com.IdClub = clu.IdClub)
WHERE (clu.nombreClub IN ('Everton', 'Estrella de Berisso'))
GROUP BY ent.IdEntrenador, ent.nombreEntrenador, ent.fechaNacimiento,
ent.direccion
HAVING (COUNT(DISTINCT clu.nombreClub) = 2)
```

(5) Listar todos los clubes en los que entrena el entrenador 'Marcos Perez'. Informar nombre del club y ciudad.

```
SELECT DISTINCT clu.IdClub, clu.nombreClub, clu.ciudad
FROM Entrenamiento e
INNER JOIN Entrenador ent ON (e.IdEntrenador = ent.IdEntrenador)
INNER JOIN Cancha c ON (e.IdCancha = c.IdCancha)
INNER JOIN Complejo com ON (c.IdComplejo = com.IdComplejo)
INNER JOIN Club clu ON (com.IdClub = clu.IdClub)
WHERE (ent.nombreEntrenador = 'Marcos Perez')
```

(6) Eliminar los entrenamientos del entrenador 'Juan Perez'.

```
DELETE FROM Entrenamiento e
WHERE (e.IdEntrenador IN (
    SELECT ent.IdEntrenador
    FROM Entrenador ent
    WHERE (ent.nombreEntrenador = 'Juan Perez')
))
```