# Practica 4

 Algunos ejercicios fueron corregidos pero no acomodados aquí, cualquier consulta, contactarse conmigo.

## Ejercicio 1

TABLAS:

***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)***

4)

Consigna:

> 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.

Solucion:

```
SELECT P.nombreP, descripción, precio, stock
FROM Producto
WHERE idProducto NOT IN
(
SELECT idProducto
FROM Producto INNER JOIN Detalle ON (Producto.idProducto = Detalle.idP
roducto)
INNER JOIN Factura ON (Factura.nroTicket = Detalle.nroTicket) INNER JOIN
Cliente
```

```
ON (Cliente.idCliente = Factura.idCliente)
WHERE telefono LIKE "221%"
)
ORDER BY nombreP
```

5)

Consigna:

> Listar para cada producto nombre, descripción, precio y cuantas veces fue vendido. Tenga en
> cuenta que puede no haberse vendido nunca el producto.

Solucion:

```
SELECT nombre, descripción, precio, SUM(Detalle.cantidad)
FROM Producto LEFT JOIN Detalle ON (Detalle.idProducto = Producto.idPr
oducto)
GROUP BY (idProducto, nombre, descripción, precio)
```

6)

Consigna:

> 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’.

Solucion:

```
SELECT nombre, apellido, DNI, teléfono, dirección
FROM Producto INNER JOIN Detalle ON (Producto.idProducto = Detalle.idP
roducto)
INNER JOIN Factura ON (Factura.nroTicket = Detalle.nroTicket) INNER JOIN
Cliente
ON (Cliente.idCliente = Factura.idCliente)
WHERE nombreP = "prod1" AND idCliente IN (
   SELECT idCliente
   FROM Producto INNER JOIN Detalle ON (Producto.idProducto = Detalle.i
```

```
dProducto)
   INNER JOIN Factura ON (Factura.nroTicket = Detalle.nroTicket) INNER J
OIN Cliente
   ON (Cliente.idCliente = Factura.idCliente)
   WHERE nombreP = "prod2"
)
AND idCliente NOT IN (
   SELECT idCliente
   FROM Producto INNER JOIN Detalle ON (Producto.idProducto = Detalle.i
dProducto)
   INNER JOIN Factura ON (Factura.nroTicket = Detalle.nroTicket) INNER J
OIN Cliente
   ON (Cliente.idCliente = Factura.idCliente)
   WHERE nombreP = "prod3"
)
```

7)

Consigna:

> 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 2019.

Solución:

```
SELECT Factura.nroTicket, total, fecha, hora, DNI
FROM Producto INNER JOIN Detalle ON (Producto.idProducto = Detalle.idP
roducto)
INNER JOIN Factura ON (Factura.nroTicket = Detalle.nroTicket) INNER JOIN
Cliente
ON (Cliente.idCliente = Factura.idCliente)
WHERE nombreP = "prod38" OR (fecha BETWEEN  1/1/2019 and 31/12/201
9)
```

8)

Consigna:

Agregar un cliente con los siguientes datos: nombre:’Jorge Luis’, apellido:’Castor’, DNI:
40578999, teléfono: ‘221-4400789’, dirección:’11 entre 500 y 501 nro:2587’ y el id de cliente:
500002. Se supone que el idCliente 500002 no existe.

Solución:

```
INSERT INTO Cliente (nombre, apellido, DNI, telefono, direccion)
VALUES ("Jorge Luis", "Castor", 40578999, "221-4400789", "11 entre 500 y 5001 nro 2857")
```

9)

Consigna:

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

Solución:

```
SELECT Factura.nroTicket, total, fecha, hora, DNI
FROM Producto INNER JOIN Detalle ON (Producto.idProducto = Detalle.idProducto)
INNER JOIN Factura ON (Factura.nroTicket = Detalle.nroTicket) INNER JOIN Cliente
ON (Cliente.idCliente = Factura.idCliente)
WHERE nombreP <> "Z" AND nombre = "Jorge" AND apellido = "Perez"
EXCEPT (
SELECT Factura.nroTicket, total, fecha, hora, DNI
FROM Producto INNER JOIN Detalle ON (Producto.idProducto = Detalle.idProducto)
INNER JOIN Factura ON (Factura.nroTicket = Detalle.nroTicket) INNER JOIN Cliente
ON (Cliente.idCliente = Factura.idCliente)
WHERE nombreP = "Z" AND nombre = "Jorge" AND apellido = "Perez"
)
```

10)

Consigna:

> Listar DNI, apellido y nombre de clientes donde el monto total comprado, teniendo en cuenta
> todas sus facturas, supere $10.000.000.

Solución:

```
SELECT DNI, apellido, nombre, SUM(Factura.total) as monto
FROM Factura INNER JOIN Cliente ON(Cliente.idCliente = Factura.idClient
e)
GROUP BY idCliente, DNI, apellido, nombre
HAVING SUM(Factura.total) > 10.000.000
```

## Ejercicio 2

TABLAS:

**Localidad = (codigoPostal, nombreL, descripcion, #habitantes)**
**Arbol = (nroArbol, especie, años, calle, nro, codigoPostal(fk))**
**Podador = (DNI, nombre, apellido, telefono, fnac, codigoPostalVive(fk))**
**Poda = (codPoda, fecha, DNI(fk), nroArbol(fk))**

1)

Consigna:

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

Solución:

```
SELECT especie, arbol.años, nombreL, Localidad.codigoPostal
FROM Arbol INNER JOIN Localidad ON (Arbol.codigoPostal = Localidad.co
digoPostal)
INNER JOIN Poda ON (Poda.nroArbol = Arbol.nroArbol) INNER JOIN Podad
or
ON (Podador.DNI = Poda.DNI))
WHERE nombre = "Jose" AND apellido = "García"
INTERSECT
(SELECT especie, arbol.años, nombreL, Localidad.codigoPostal
FROM Arbol INNER JOIN Localidad ON (Arbol.codigoPostal = Localidad.co
digoPostal)
INNER JOIN Poda ON (Poda.nroArbol = Arbol.nroArbol) INNER JOIN Podad
or
ON (Podador.DNI = Poda.DNI))
WHERE nombre = "Juan" AND apellido = "Perez")
```

2)

Consigna:

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

Solución:

```
SELECT DNI, nombre, apellido, fnac, Localidad.nombreL
FROM Podador INNER JOIN Poda ON (Poda.DNI = Podador.DNI) INNER JOI
N
Arbol ON (Arbol.nroArbol = Poda.nroArbol) INNER JOIN Localidad
ON (Localidad.codigoPostal = Arbol.codigoPostal)
WHERE (Poda.fecha BETWEEN 1/1/2023 and 31/12/2023)
```

3)

Consigna:

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

Solución:

```
SELECT especie, años, calle, nro, Localidad.nombreL
FROM Arbol INNER JOIN Localidad ON (Arbol.codigoPostal = Localidad.co
digoPostal)
WHERE Arbol.nroArbol NOT IN (SELECT nroArbol
                                        FROM Poda)
```

4)

Consigna:

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

Solucion:

```
SELECT especie, años, Arbol.calle, Arbol.nro, Localidad.nombreL
FROM Arbol INNER JOIN Poda ON (Poda.nroArbol = Arbol.nroArbol)
INNER JOIN Localidad ON (Localidad.codigoPostal = Arbol.codigoPostal)
WHERE (Poda.fecha BETWEEN 1/1/2022 AND 31/12/2022)
EXCEPT(
SELECT especie, años, Arbol.calle, Arbol.nro, Localidad.nombreL
FROM Arbol INNER JOIN Poda ON (Poda.nroArbol = Arbol.nroArbol)
INNER JOIN Localidad ON (Localidad.codigoPostal = Arbol.codigoPostal)
WHERE (Poda.fecha BETWEEN 1/1/2023 AND 31/12/2023)
)
```

5)

Consigna:

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 DNI, nombre, apellido, fnac, l.nombreL
FROM Podador pod INNER JOIN Localidad l ON (l.codigoPostal = pod.codi
goPostalVive)
WHERE (Podador.nombre LIKE "%ata") AND EXISTS (
   SELECT *
   FROM Poda p
   WHERE (fecha BETWEEN 1/1/2024 and 31/12/2024) and p.DNI = Pod.DNI
)
```

6)

Consigna:

> Listar DNI, apellido, nombre, teléfono y fecha de nacimiento de podadores que solo podaron
> árboles de especie ‘Coníferas’.

Solución:

```
SELECT DNI, apellido, nombre, telefono, fnac
FROM Arbol INNER JOIN Poda ON (Poda.nroArbol = Arbol.nroArbol) INNER
JOIN
Podador ON (Podador.DNI = Poda.DNI)
WHERE (Arbol.especie = "Coniferas")
EXCEPT (
SELECT DNI, apellido, nombre, telefono, fnac
FROM Arbol INNER JOIN Poda ON (Poda.nroArbol = Arbol.nroArbol) INNER
JOIN
Podador ON (Podador.DNI = Poda.DNI)
WHERE (Arbol.especie <> "Coniferas")
)
```

7)

Consigna:

> Listar especies de árboles que se encuentren en la localidad de ‘La Plata’ y también en la

localidad de ‘Salta’.

Solución:

```
SELECT especie
FROM Arbol INNER JOIN Localidad ON (Localidad.codigoPostal = Arbol.co
digoPostal)
WHERE (Localidad.nombreL = "Salta")
INTERSECT(
SELECT especie
FROM Arbol INNER JOIN Localidad ON (Localidad.codigoPostal = Arbol.co
digoPostal)
WHERE (Localidad.nombreL = "La Plata")
)
```

8)

Consigna:

Eliminar el podador con DNI 22234566.

Solucion:

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

9)

Consigna:

Reportar nombre, descripción y cantidad de habitantes de localidades que tengan menos de 100
árboles.

Solucion:

```
SELECT nombreL, descripcion, #habitantes
FROM Localidad INNER JOIN Arbol ON (Arbol.codigoPostal = Localidad.co
digoPostal)
GROUP BY nombreL, descripcion, #habitantes, codigoPostal
HAVING COUNT(nroArbol) < 100
```

## Ejercicio 3

TABLAS:

**Banda = (codigoB, nombreBanda, genero_musical, año_creacion)**
**Integrante = (DNI, nombre, apellido, dirección, email, fecha_nacimiento, codigoB(fk))**
**Escenario = (nroEscenario, nombre_escenario, ubicación, cubierto, m2, descripción)**
**Recital = (fecha, hora, nroEscenario (fk), codigoB (fk))**

1)

Consigna:

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

Solución:

```
SELECT DNI, nombre, apellido, direccion, email
FROM Integrante INNER JOIN Recital ON (Integrante.codigoB = Recital.codi
goB)
WHERE (Integrante.fecha_nacimiento BETWEEN 1/1/1980 AND 31/12/1990)
AND (Recital.fecha BETWEEN 1/1/2023 AND  31/12/2023)
```

2)

Consigna:

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

Solución:

```
SELECT Banda.nombre, genero_musical, año_creacion
FROM Banda INNER JOIN Recital ON (Recital.codigoB = Banda.codigoB)
WHERE (Recital.fecha BETWEEN 1/1/2023 AND 31/12/2023)
EXCEPT(
```

```
SELECT Banda.nombre, genero_musical, año_creacion
FROM Banda INNER JOIN Recital ON (Recital.codigoB = Banda.codigoB)
WHERE (Recital.fecha BETWEEN 1/1/2022 AND 31/12/2022)
)
```

3)

Consigna:

> 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.

Solución:

```
SELECT Banda.nombreBanda, Recital.fecha, Recital.hora, Escenario.nombr
e_escenario,
Escenario.ubicacion
FROM Banda INNER JOIN Recital ON (Banda.codigoB = Recital.codigoB) IN
NER JOIN
Escenario ON (Escenario.nroEscenario = Recital.nroEscenario)
WHERE Recital.fecha = 4/12/2023
```

4)

Consigna:

> 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’.

Solución:

```
SELECT DNI, nombre, apellido, email
FROM Integrante INNER JOIN Recital ON (Integrante.codigoB = Recital.codi
goB)
INNER JOIN Escenario ON (Escenario.nroEscenario = Recital.nroEscenario)
WHERE nombre_escenario = "Gustavo Cerati"
INTERSECT (
```

```
SELECT DNI, nombre, apellido, email
FROM Integrante INNER JOIN Recital ON (Integrante.codigoB = Recital.codi
goB)
INNER JOIN Escenario ON (Escenario.nroEscenario = Recital.nroEscenario)
WHERE nombre_escenario = "Carlos Gardel" )
```

5)

Consigna:

> Reportar nombre, género musical y año de creación de bandas que tengan más de 8 integrantes.

Solución:

```
SELECT nombreBanda, genero_musical, año_creacion
FROM Banda INNER JOIN Integrante ON (Integrante.codigoB = Banda.codi
goB)
GROUP BY Banda.codigoB, nombreBanda, genero_musical, año_creacion
HAVING COUNT(*) > 8
```

6)

Consigna:

> Listar nombre de escenario, ubicación y descripción de escenarios que solo tuvieron recitales
> con el género musical rock and roll. Ordenar por nombre de escenario.

Solución:

```
SELECT nombre_escenario, ubicación, descripción
FROM Escenario e INNER JOIN Recital r ON (e.nroEscenario = r.nroEscenar
io)
INNER JOIN Banda b ON (b.codigoB = r.codigoB)
WHERE genero_musical = "rock and roll" AND e.nroEscenario NOT IN(
SELECT e.nroEscenario
FROM Escenario e INNER JOIN Recital ON (e.nroEscenario = Recital.nroEsc
enario)
INNER JOIN Banda ON (Banda.codigoB = Recital.codigoB)
```

```
WHERE genero_musical <> "rock and roll"
)
ORDER BY e.nombre_escenario
```

7)

Consigna:

> 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

Solución:

```
SELECT nombre_banda, genero_musical, año_creacion
FROM Banda INNER JOIN Recital ON (Banda.codigoB = Recital.codigoB)
WHERE (fecha BETWEEN 1/1/2023 AND 31/12/2023) AND cubierto = true
```

8)

Consigna:

> Reportar para cada escenario, nombre del escenario y cantidad de recitales durante 2024.

Solución:

```
SELECT nombre_escenario, COUNT(*)
FROM Escenario INNER JOIN Recital ON (Recital.nro_escenario = Escenari
o.nro_escenario)
WHERE fecha BETWEEN 1/1/2024 AND 31/12/2024
GROUP BY nro_escenario, nombre_escenario
```

9)

Consigna:

> Modificar el nombre de la banda ‘Mempis la Blusera’ a: ‘Memphis la Blusera’.

Solución:

```
UPDATE Banda SET nombreBanda = "Memphis la Blusera" WHERE nombre
Banda = "Mempis la Blusera"
```

## Ejercicio 4

TABLAS:

***PERSONA = (DNI, Apellido, Nombre, Fecha_Nacimiento, Estado_Civil, Genero)***
***ALUMNO = (DNI (fk), Legajo, Año_Ingreso)***
***PROFESOR = (DNI (fk), Matricula, Nro_Expediente)***
***TITULO = (Cod_Titulo, Nombre, Descripción)***
***TITULO-PROFESOR = (Cod_Titulo (fk), DNI (fk), Fecha)***
***CURSO = (Cod_Curso, Nombre, Descripción, Fecha_Creacion, Duracion)***
***ALUMNO-CURSO = (DNI (fk), Cod_Curso (fk), Año, Desempeño, Calificación)***
***PROFESOR-CURSO = (DNI (fk), Cod_Curso (fk), Fecha_Desde, Fecha_Hasta)***

1)

Consigna:

> Listar DNI, legajo y apellido y nombre de todos los alumnos que tengan año de ingreso inferior a
> 2014.

Solución:

```
SELECT ALUMNO.DNI, Legajo, Apellido, Nombre
FROM ALUMNO INNER JOIN PERSONA ON (ALUMNO.DNI = PERSONA.DN
I)
WHERE Año_Ingreso < 1/1/2014
```

2)

Consigna:

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

Solución:

```
SELECT PROFESOR.DNI, Matrícula, Apellido, Nombre
FROM PROFESOR INNER JOIN PROFESOR-CURSO ON (PROFESOR.DNI = P
ROFESOR-CURSO.DNI)
INNER JOIN CURSO ON (CURSO.Cod_curso = PROFESOR-CURSO.Cod_cur
so)
WHERE (CURSO.Duracion > 100)
ORDER BY PROFESOR.DNI
```

3)

Consigna:

Listar el DNI, Apellido, Nombre, Género y Fecha de nacimiento de los alumnos inscriptos al
curso con nombre “Diseño de Bases de Datos” en 2023.

Solución:

```
SELECT  ALUMNO.DNI, Apellido, Nombre, Genero, Fecha_nacimiento
FROM ALUMNO INNER JOIN PERSONA ON (PERSONA.DNI = ALUMNO.DN
I)
INNER JOIN ALUMNO-CURSO ON (ALUMNO.DNI = ALUMNO-CURSO.DNI)
WHERE ALUMNO-CURSO.Año = 2023 AND ALUMNO.CURSO.Cod_curso IN
(
SELECT Cod_curso
FROM CURSO
WHERE Nombre = "Diseño de Bases de Datos"
)
```

4)

Consigna:

Listar el DNI, Apellido, Nombre y Calificación de aquellos alumnos que obtuvieron una
calificación superior a 8 en algún curso que dicta el profesor “Juan Garcia”. Dicho listado deberá
estar ordenado por Apellido y nombre.

Solución:

```
SELECT DNI, Apellido, Nombre, Calificación
FROM ALUMNO INNER JOIN PERSONA ON (ALUMNO.DNI = PERSONA.DN
I)
INNER JOIN ALUMNO-CURSO ON (ALUMNO.DNI = ALUMNO-CURSO.DNI)
INNER JOIN PROFESOR-CURSO ON (PROFESOR-CURSO.Cod_curso = ALU
MNO-CURSO.Cod_curso)
WHERE Calificación > 8 AND PROFESOR-CURSO.DNI IN (
SELECT DNI
FROM PROFESOR
WHERE Nombre = "Juan" AND Apellido = "García"
)
ORDER BY PERSONA.Apellido, PERSONA.Nombre
```

5)

Consigna:

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

Solución:

```
SELECT PROFESOR.DNI, Apellido, Nombre, Matrícula
FROM PROFESOR INNER JOIN PROFESOR-TITULO ON (PROFESOR.DNI = P
ROFESOR-TITULO.DNI)
INNER JOIN PERSONA ON (PERSONA.DNI = PROFESOR.DNI)
GROUP BY PROFESOR.DNI, Apellido, Nombre, Matrícula
HAVING COUNT(PROFESOR-TITULO.Cod_titulo) > 3
ORDER BY Apellido, Nombre
```

6)

Consigna:

> Listar el 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.

Solución:

```
SELECT PROFESOR.DNI, Apellido, Nombre, SUM(CURSO.DURACION), AVG
(CURSO.DURACION)
FROM PROFESOR INNER JOIN PERSONA ON (PERSONA.DNI = PROFESOR.
DNI)
INNER JOIN PROFESOR-CURSO ON (PROFESOR-CURSO.DNI = PROFESO
R.DNI)
INNER JOIN CURSO ON (PROFESOR-CURSO.Cod_curso = CURSO.Cod_cur
so)
GROUP BY PROFESOR.DNI, Apellido, Nombre
```

7)

Consigna:

> Listar Nombre y Descripción del curso que posea más alumnos inscriptos y del que posea
> menos alumnos inscriptos durante 2024.

Solución:

```
SELECT Nombre, Descripción
FROM ALUMNO-CURSO INNER JOIN ALUMNO ON (ALUMNO.DNI = ALUM
NO-CURSO.DNI)
WHERE Año = 2024
GROUP BY Cod_curso
HAVING COUNT(ALUMNO.DNI) >= ALL(
        SELECT COUNT(*)
        FROM ALUMNO_CURSO
        WHERE Año = 2024
        GROUP BY Cod_curso
```

```
)
OR COUNT(ALUMNO.DNI) <= ALL(
        SELECT COUNT(*)
        FROM ALUMNO_CURSO
        WHERE Año = 2024
        GROUP BY Cod_curso
)
```

8)

Consigna:

> 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.

Solución:

```
SELECT DNI, Apellido, Nombre, Legajo
FROM ALUMNO INNER JOIN PERSONA ON (PERSONA.DNI = ALUMNO.DN
I)
INNER JOIN ALUMNO-CURSO ON (ALUMNO.DNI = ALUMNO-CURSO.DNI)
WHERE Cod_curso IN (
   SELECT Cod_curso
   FROM CURSO
   WHERE Nombre LIKE "%BD%"
)
AND Cod_curso NOT IN(
   SELECT Cod_curso
   FROM ALUMNO-CURSO
   WHERE AÑO = 2023
)
AND ALUMNO-CURSO.Año = 2022
```

9)

Consigna:

> Agregar un profesor con los datos que prefiera y agregarle el título con código: 25.

Solución:

```
INSERT INTO PROFESOR (DNI, Matrícula, Nro_Expediente) VALUES (44321213, "KALSA-Z2A", 242)
INSERT INTO TITULO-PROFESOR (Cod_titulo, DNI, Fecha) VALUES (25,44321213, 1/11/2024)
```

10)

Consigna:

> Modificar el estado civil del alumno cuyo legajo es ‘2020/09’, el nuevo estado civil es divorciado.

Solución:

```
UPDATE PERSONA
SET Estado_Civil = "divorciado"
FROM PERSONA INNER JOIN ALUMNO ON (ALUMNO.DNI = PERSONA.DNI)
WHERE Legajo = "2020/09"
```

11)

Consigna:

> Dar de baja el alumno con DNI 30568989. Realizar todas las bajas necesarias para no dejar el
> conjunto de relaciones en un estado inconsistente

Solución:

```
DELETE FROM ALUMNO-CURSO WHERE DNI = 30568989
DELETE FROM ALUMNO WHERE DNI = 30568989
DELETE FROM PERSONA WHERE DNI = 30568989
```

## Ejercicio 5

TABLAS:

**AGENCIA = (RAZON_SOCIAL, dirección, telef, e-mail)**
**CIUDAD = (CODIGOPOSTAL, nombreCiudad, añoCreación)**
**CLIENTE = (DNI, nombre, apellido, teléfono, dirección)**
**VIAJE = (FECHA, HORA, DNI (fk), cpOrigen(fk), cpDestino(fk), razon_social(fk), descripcion)**
//cpOrigen y cpDestino corresponden a la ciudades origen y destino del viaje

💡 Consultar ejercicio 6 y 8. Las soluciones no las termino de entender del todo

1)

Consigna:

> 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.

Solución:

```
SELECT RAZON_SOCIAL, dirección, telef
FROM AGENCIA INNER JOIN VIAJE ON (VIAJE.razon_social = AGENCIA.RA
ZON_SOCIAL)
INNER JOIN CLIENTE ON (CLIENTE.DNI = VIAJE.DNI)
WHERE apellido = "Roma" AND VIAJE.cpOrigen IN (
SELECT CODIGOPOSTAL
FROM CIUDAD
WHERE nombreCiudad = "La Plata"
```

```
)
ORDER BY AGENCIA.RAZON_SOCIAL, telef
```

2)

Consigna:

> 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’.

Solución:

```
SELECT FECHA, HORA, CLIENTE.DNI, nombre, apellido, nombreCiudad
FROM CLIENTE INNER JOIN VIAJE ON (VIAJE.DNI = CLIENTE.DNI)
INNER JOIN CIUDAD ON (CIUDAD.CODIGOPOSTAL = cpOrigen OR
CIUDAD.CODIGOPOSTAL = cpDestino)
WHERE FECHA BETWEEN (1/1/2019 AND 31/12/2019) AND descripcion LIKE
'%demorado%'
```

3)

Consigna:

> Reportar información de agencias que realizaron viajes durante 2019 o que tengan dirección de
> mail que termine con ‘@jmail.com’.

Solución:

```
SELECT RAZON_SOCIAL, dirección, telef, e-mail
FROM AGENCIA INNER JOIN VIAJE ON (VIAJE.razon_social = AGENCIA.RA
ZON_SOCIAL)
WHERE FECHA BETWEEN (1/1/2019 AND 31/12/2019) OR e-mail LIKE '%@j
mail.com'
```

4)

**AGENCIA = (RAZON_SOCIAL, dirección, telef, e-mail)**
**CIUDAD = (CODIGOPOSTAL, nombreCiudad, añoCreación)**
**CLIENTE = (DNI, nombre, apellido, teléfono, dirección)**
**VIAJE = (FECHA, HORA, DNI (fk), cpOrigen(fk), cpDestino(fk), razon_social(fk), descripcion)**
//cpOrigen y cpDestino corresponden a la ciudades origen y destino del viaje

Consigna:

Listar datos personales de clientes que viajaron solo con destino a la ciudad de ‘Coronel
Brandsen

Solución:

```
SELECT
   C.DNI,
   C.nombre,
   C.apellido
FROM
   CLIENTE C
INNER JOIN
   VIAJE V ON C.DNI = V.DNI  -- Une Cliente con sus Viajes
INNER JOIN
   CIUDAD Destino ON V.cpDestino = Destino.CODIGOPOSTAL -- Une Viaje
con la Ciudad Destino
WHERE
   Destino.nombreCiudad = 'Coronel Brandsen'
   AND C.DNI NOT IN (
      SELECT
         V2.DNI
      FROM
         VIAJE V2
      INNER JOIN
         CIUDAD Destino2 ON V2.cpDestino = Destino2.CODIGOPOSTAL
      WHERE
         Destino2.nombreCiudad <> 'Coronel Brandsen'
   )
```

5)

Consigna:

> Informar cantidad de viajes de la agencia con razón social ‘TAXI Y’ realizados a ‘Villa Elisa’.

Solución:

```
SELECT COUNT(*)
FROM AGENCIA INNER JOIN VIAJE ON (VIAJE.razon_social = AGENCIA.RA
ZON_SOCIAL)
WHERE AGENCIA.RAZON_SOCIAL = 'TAXI Y' AND cpDestino IN (
SELECT CODIGOPOSTAL
FROM CIUDAD
WHERE nombreCiudad = 'Villa Elisa'
)
```

6)

**AGENCIA = (RAZON_SOCIAL, dirección, telef, e-mail)**
**CIUDAD = (CODIGOPOSTAL, nombreCiudad, añoCreación)**
**CLIENTE = (DNI, nombre, apellido, teléfono, dirección)**
**VIAJE = (FECHA, HORA, DNI (fk), cpOrigen(fk), cpDestino(fk), razon_social(fk), descripcion)**
//cpOrigen y cpDestino corresponden a la ciudades origen y destino del viaje

Consigna:

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

Solución 1 :

"Seleccionar los clientes (C) para los cuales **no existe** una agencia (A) con la que el cliente **no haya viajado**."

```
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 V.DNI = C.DNI
        AND V.razon_social = A.RAZON_SOCIAL
    )
)
```

Solución 2:

```
SELECT C.nombre, C.apellido, C.direccion, C.telefono
FROM CLIENTE C INNER JOIN VIAJE v ON (v.DNI = C.DNI)
GROUP BY C.DNI, C.nombre, C.apellido, C.direccion, C.telefono
HAVING
    COUNT(DISTINCT V.razon_social) = ( -- Contamos cuántas agencias dist
intas usó ese cliente
        SELECT COUNT(*)
        FROM AGENCIA -- Comparamos con el número total de agencias en la
BD
    );
```

7)

Consigna:

> Modificar el cliente con DNI 38495444 actualizando el teléfono a ‘221-4400897’.

Solución:

```
UPDATE CLIENTE SET telefono='221-4400897' WHERE DNI = 38495444
```

8)

Consigna:

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

Solución:

```
SELECT VIAJE.RAZON_SOCIAL, direccion, telef
FROM VIAJE
GROUP BY viaje.razon_social, direccion, telef
HAVING COUNT(*) >= ALL (
   SELECT COUNT(*)
   FROM VIAJE
   GROUP BY viaje.razon_social
)
```

```
SELECT
   A.RAZON_SOCIAL,
   A.direccion,
   A.telef
FROM
   AGENCIA A
INNER JOIN
   VIAJE V ON A.RAZON_SOCIAL = V.razon_social
GROUP BY
   A.RAZON_SOCIAL, A.direccion, A.telef -- Agrupamos por los datos de sal
ida
HAVING
   COUNT(*) = (
      -- Subconsulta: Encuentra el valor máximo de viajes realizados por CU
ALQUIER agencia
      SELECT
         MAX(Viajes_Realizados)
      FROM
         (
            SELECT
               COUNT(*) AS Viajes_Realizados
            FROM
               VIAJE
```

```
            GROUP BY
                razon_social
        ) AS ConteoViajes
    );
```

9)

Consigna:

> Reportar nombre, apellido, dirección y teléfono de clientes con al menos 10 viajes.

Solución:

```
SELECT nombre, apellido, dirección, teléfono
FROM CLIENTE INNER JOIN VIAJE ON (CLIENTE.DNI = VIAJE.DNI)
GROUP BY CLIENTE.DNI, nombre, apellido, dirección, teléfono
HAVING COUNT(CLIENTE.DNI) >= 10
```

10)

Consigna:

> Borrar al cliente con DNI 40325692.

Solución:

```
DELETE FROM VIAJE WHERE DNI = 40325692
DELETE FROM CLIENTE WHERE DNI = 40325692
```

## Ejercicio 6

TABLAS:

***Técnico = (codTec, nombre, especialidad)*** *// técnicos*
***Repuesto = (codRep, nombre, stock, precio)*** *// repuestos*
***RepuestoReparacion = (nroReparac (fk), codRep (fk), cantidad, precio)*** *// repuestos utilizados en reparaciones.*
***Reparación (nroReparac, codTec (fk), precio_total, fecha)*** *// reparaciones realizadas.*

💡 Consultar ejercicio 6
Principalmente porque no sé como sacar máximo y mínimo cuando se trata de algo que hay que contabilizar previamente

1)

**Consigna:**

Listar los repuestos, informando el nombre, stock y precio. Ordenar el resultado por precio

**Solución:**

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

2)

**Consigna:**

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’.

**Solución/es:**

```
SELECT nombre, stock, precio
FROM Repuesto
WHERE codRep IN (
SELECT codRep
FROM RepuestoReparacion INNER JOIN
Reparación ON (RepuestoReparacion.nroReparac = Reparación.nroRepara
c)
WHERE fecha BETWEEN (1/1/2023 AND 31/12/2023) AND codTec NOT IN (
        SELECT codTec
        FROM Técnico
        WHERE nombre = 'José' AND apellido = 'Gonzales'
```

```
        )
)
```

Solucion 2:

```
SELECT nombre, stock, precio
FROM Repuesto INNER JOIN RepuestoReparacion ON
(Repuesto.codRep = RepuestoReparacion.codRep)
INNER JOIN Reparación ON (RepuestoReparacion.nroReparac = Reparació
n.nroReparac)
WHERE fecha BETWEEN (1/1/2023 AND 31/12/2023) AND codTec NOT IN (
        SELECT codTec
        FROM Técnico
        WHERE nombre = 'José' AND apellido = 'Gonzales'
    )
)
```

3)

Consigna:

> Listar el nombre y especialidad de técnicos que no participaron en ninguna reparación. Ordenar
> por nombre ascendentemente.

Solución

```
SELECT nombre, especialidad
FROM Técnico
WHERE codTec NOT IN (
SELECT codTec
From Reparación
)
ORDER BY nombre ASC
```

4)

Consigna:

> Listar el nombre y especialidad de los técnicos que solamente participaron en reparaciones
> durante 2022.

Solución:

```
SELECT nombre, especialidad
FROM Técnico INNER JOIN Reparación ON (Tecnico.codTec = Reparación.
codTec)
WHERE Fecha BETWEEN (1/1/2022 AND 31/12/2022) AND codTec NOT IN (
   SELECT codTec
   FROM Reparación
   WHERE Fecha < 1/1/2022 OR fecha > 31/12/2022
)
```

5)

Consigna:

> 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.

Solución:

```
SELECT nombre, stock, COUNT(DISTINCT Reparación.codTec)
FROM Repuesto LEFT JOIN RepuestoReparación ON (Repuesto.codRep = R
epuestoReparación.codRep)
LEFT JOIN Reparación ON (Reparación.codReparac = RepuestoReparació
n.codReparac)
GROUP BY Repuesto.codRep, nombre, stock
```

6)

Consigna:

> Listar nombre y especialidad del técnico con mayor cantidad de reparaciones realizadas y el
> técnico con menor cantidad de reparaciones

Solución:

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

7)

Consigna:

> 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 $10000.

Solución:

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

8)

Consigna:

> Proyectar número, fecha y precio total de aquellas reparaciones donde se utilizó algún repuesto
> con precio en el momento de la reparación mayor a $10000 y menor a $15000.

Solución:

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

9)

> 💡 A preguntar

Consigna:

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

Solución:

```
SELECT nombre, stock, precio
FROM Repuesto r
WHERE NOT EXISTS (
   SELECT *
      FROM RepuestoReparacion repa
      WHERE repa.codTec = rep.codTec
   WHERE NOT EXISTS (
      SELECT Reparacion
```

```
    )
)
```

10)

Consigna:

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

Solucion:

```
SELECT fecha, codTec, precio_total
FROM Reparacion
WHERE nroReparac IN (
SELECT nroReparac
FROM RepuestoReparacion
GROUP BY nroReparac
HAVING COUNT(DISTINCT codRep) >= 10
)
```

## 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)

Consigna:

> Reportar nombre y año de fundación de aquellos clubes de la ciudad de La Plata que no poseen

estadio.

Solucion:

```
SELECT Club.nombre, Club.anioFundacion
FROM Club
WHERE codigoClub NOT IN (
    SELECT codigoClub
    FROM Estadio
)
AND codigoCiudad IN (
    SELECT codigoCiudad
    FROM Ciudad
    WHERE nombre = 'La Plata'
)
```

2)

Consigna:

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

Solución:

```
SELECT Club.nombre
FROM Club
WHERE codigoClub NOT IN (
        SELECT codigoClub
        FROM ClubJugador INNER JOIN Jugador ON (ClubJugador.DNI = Jug
ador.DNI)
        WHERE Jugador.codigoCiudad IN (
                SELECT codigoCiudad
                FROM Ciudad
                WHERE nombre = 'Berisso'
            )
)
```

3)

Consigna:

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

Solución:

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

4)

Consigna:

> 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

Solución:

```
SELECT DNI, nombre, apellido
FROM Jugador
WHERE edad > 29 AND DNI IN (
        SELECT DNI
        FROM ClubJugador INNER JOIN Club ON (Club.codigoClub = ClubJ
ugador.codigoClub)
        INNER JOIN Ciudad ON (Club.codigoCiudad = Ciudad.codigoCiuda
d)
        WHERE Ciudad.nombre = 'Córdoba'
)
```

5)

Consigna:

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

Solución:

```
SELECT Club.nombre, AVG(Jugador.edad)
FROM Club INNER JOIN ClubJugador ON (Club.codigoClub = ClubJugador.
codigoClub)
INNER JOIN Jugador ON (Jugador.DNI = ClubJugador.DNI)
WHERE Jugador.DNI IN (
   SELECT DNI
   FROM ClubJugador
   WHERE hasta > 4/11/2024 #Es la fecha actual
)
GROUP BY Club.codigoClub, Club.nombre
```

6)

Consigna:

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

Solución:

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

7)

Consigna:

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

Solucion:

```
SELECT Club.nombre
FROM Club INNER JOIN ClubJugador ON (Club.codigoClub = ClubJugador.
codigoClub)
WHERE DNI NOT IN (
   SELECT DNI
   FROM Jugador INNER JOIN Ciudad ON (Jugador.codigoCiudad = Ciuda
d.codigoCiudad)
   WHERE Ciudad.nombre = 'Mar Del Plata'
)
```

8)

Consigna:

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

Solución:

```
SELECT nombre, apellido
FROM Jugador
WHERE NOT EXISTS (
      SELECT *
      FROM Club
      WHERE NOT EXISTS (
            SELECT *
            FROM ClubJugador
            WHERE ClubJugador.codigoClub  = Club.codigoClub AND
            ClubJugador.DNI = Jugador.DNI AND Club.codigoCiudad IN (
               SELECT codigoCiudad
               FROM Ciudad
               WHERE nombre = 'Córdoba'
            )
```

```
        )
  )
```

9)

Consigna:

> Agregar el club “Estrella de Berisso”, con código 1234, que se fundó en 1921 y que pertenece a
> la ciudad de Berisso. Puede asumir que el codigoClub 1234 no existe en la tabla Club.

Solución:

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

## Ejercicio 8

TABLAS:

**Equipo =** (codigoE, nombreE, descripcionE)
**Integrante =** (DNI, nombre, apellido,ciudad,email, telefono,codigoE(fk))
**Laguna =** (nroLaguna, nombreL, ubicación,extension, descripción)
**TorneoPesca =** (codTorneo, fecha,hora, nroLaguna(fk), descripcion)
**Inscripcion =** (codTorneo(fk),codigoE(fk), asistio, gano) // asistio y gano son true o false según corresponda

1)

Consigna:

> Listar DNI, nombre, apellido y email de integrantes que sean de la ciudad ‘La Plata’ y estén
> inscriptos en torneos disputados en 2023.

```
SELECT DNI, nombre, apellido, email
FROM Integrante
WHERE ciudad = 'La Plata' AND CodigoE IN (
SELECT CodigoE
FROM Inscripcion INNER JOIN TorneoPesca ON (TorneoPesca.codTorneo
= Inscripcion.codTorneo)
WHERE fecha BETWEEN 1/1/2023 AND 31/12/2023
)
```

2)

Consigna:

> Reportar nombre y descripción de equipos que solo se hayan inscripto en torneos de 2020.

Solución:

```
SELECT nombreE, descripcionE
FROM Equipo
where codigoE NOT IN (
SELECT codigoE
FROM Inscripcion INNER JOIN TorneoPesca ON (TorneoPesca.codTorneo
= Inscripcion.codTorneo)
where fecha < 1/1/2020 OR fecha > 31/12/2020
)
```

3)

Consigna:

> 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.

Solución:

```
SELECT DNI, nombre, apellido, email, ciudad
FROM Integrante INNER JOIN Inscripcion ON (Integrante.codigoE = Inscipci
on.codigoE)
INNER JOIN TorneoPesca ON (TorneoPesca.codTorneo = Inscripcion.codT
orneo)
INNER JOIN Laguna ON (TorneoPesca.nroLaguna = Laguna.nroLaguna)
WHERE nombreL = 'La salada, Coronel Granada' AND codigoE NOT IN(
SELECT codigoE
FROM Inscripcion INNER JOIN TorneoPesca ON (TorneoPesca.codTorneo
= Inscripcion.codTorneo)
where fecha BETWEEN 1/1/2023 AND 31/12/2023
)
```

4)

Consigna:

> Reportar nombre y descripción de equipos que tengan al menos 5 integrantes. Ordenar por nombre

Solución

```
SELECT nombreE, descripcion
FROM Equipo INNER JOIN Integrante ON (Equipo.codigoE = Integrante.codi
goE)
GROUP BY Equipo.codigoE, nombreE, descripción
HAVING COUNT(Equipo.codigoE) >= 5
ORDER BY nombreE
```

5)

Cosigna:

> Reportar nombre y descripción de equipos que tengan inscripciones en todas las lagunas.

Solución:

```
SELECT nombreE, descripcion
FROM Equipo
WHERE codigoE IN (
   SELECT codigoE
   FROM Inscripcion ins
   WHERE NOT EXISTS (
      SELECT *
      FROM Laguna l
      WHERE NOT EXISTS (
         SELECT *
         FROM TorneoPesca TP
         WHERE l.nroLaguna = TP.roLaguna
         AND TP.codTorneo = ins.codTorneo
       )
)
)
```

6)

Consigna:

Eliminar el equipo con código 10000

Solución:

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

7)

Consigna:

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

Solución:

```
SELECT nombreL, ubicación, extension, descripción
FROM Laguna
WHERE nroLaguna NOT IN (
SELECT nroLaguna
From TorneoPesca
)
```

8)

Consigna:

> Reportar nombre y descripción de equipos que tengan inscripciones a torneos a disputarse durante 2024, pero no tienen inscripciones a torneos de 2023.
>
> 7/10/2024 Es la fecha actual en la que lo hice

```
SELECT nombreE, descripcion
FROM Equipo INNER JOIN Inscripcion Inscripcion
ON (Equipo.codigoE = Inscipcion.codigoE)
INNER JOIN TorneoPesca ON (TorneoPesca.codTorneo = Inscripcion.codT
orneo)
WHERE fecha BETWEEN (8/11/2023 AND 31/12/2023) AND codigoE NOT IN
(
   SELECT codigoE
   From Inscripcion INNER JOIN TorneoPesca
   ON (TorneoPesca.codTorneo = Inscripcion.codTorneo)
   WHERE fecha BETWEEN 1/1/2023 AND 31/12/2023
)
```

9)

Consigna:

Listar DNI, nombre, apellido, ciudad y email de integrantes que ganaron algún torneo que se disputó
en la laguna con nombre: ‘Laguna de Chascomús’.

8/11 es el día anterior al actual

Solución:

```
SELECT DNI, nombre, apellido
FROM Integrante INNER JOIN Inscripcion ON (Integrante.codigoE = Inscrip
cion.codigoE)
INNER JOIN TorneoPesca ON (Inscripcion.codTorneo = TorneoPesca.codT
orneo)
WHERE TorneoPesca.fecha < 8/11/2024 AND gano = true AND nroLaguna I
N (
   SELECT nroLaguna
   FROM Laguna
   WHERE nombre = 'Laguna de Chascomús'
)
```

## Ejercicio 9

TABLAS:

**Proyecto =** (codProyecto, nombrP,descripcion, fechaInicioP, fechaFinP, fechaFinEstimada, DNIResponsable(FK), equipoBackend(FK), equipoFrontend(FK)) // DNIResponsable corresponde a un empleado, equipoBackend y equipoFrontend corresponden a equipos
**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)

Consigna:

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.

Solución:

```
SELECT nombrP, descripcion, fechaInicioP, fechaFinP
FROM Proyecto
WHERE fechaFinEstimada < fechaFinP
```

2)

Consigna:

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.

Solución:

```
SELECT *
FROM Empleado
WHERE DNI NOT IN (
   SELECT DNIResponsable
   FROM Proyecto
)
```

3)

Consigna:

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

Solución:

```
SELECT DNI, nombre, apellido, teléfono, dirección
FROM Empleado
WHERE DNI IN (
SELECT DNILider
FROM Equipo
```

```
GROUP BY DNILider
HAVING COUNT(codEquipo) > 1
)
```

4)

Consigna:

> 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.

En este enunciado fueron propuestas 2 soluciones, principalmente porque la primera no termina de convencer si funciona de dicha forma. Fue consultado con un ayudante pero no termino de estar seguro si se puede o no

Solución:

```
SELECT DNI, nombre, apellido, telefono, direccion
FROM Empleado
WHERE DNI IN (
   SELECT DNI
   FROM Empleado_equipo eq INNER JOIN Equipo e ON (e.codEquipo = eq.
codEquipo)
   INNER JOIN Proyecto p ON (p.equipoBackend = e.codEquipo OR p.equip
oFrontend = e.codEquipo)
   WHERE p.nombrP = 'Proyecto X'
)
AND DNI NOT IN (
   SELECT DNILider
   FROM Equipo
)
AND DNI NOT IN (
   SELECT DNIResponsable
```

```
    FROM Proyecto
)
```

Solución 2:

```
SELECT DNI, nombre, apellido, telefono, direccion
FROM Empleado
WHERE DNI IN (
    SELECT DNI
    FROM Empleado_equipo eq INNER JOIN Equipo e ON (e.codEquipo = eq.
codEquipo)
    INNER JOIN Proyecto p ON (p.equipoBackend = e.codEquipo)
    INNER JOIN Equipo e2 ON (p.equipoFrontend = e2.codEquipo)
    WHERE p.nombrP = 'Proyecto X'
)
AND DNI NOT IN (
    SELECT DNILider
    FROM Equipo
)
AND DNI NOT IN (
    SELECT DNIResponsable
    FROM Proyecto
)
```

5)

Consigna:

> Listar nombre de equipo y datos personales de líderes de equipos que no tengan empleados
> asignados y trabajen con tecnología ‘Java’.

Solución:

```
SELECT nombreE, e.DNI, e.nombre, e.apellido
FROM Equipo INNER JOIN Empleado e ON (e.DNI = Equipo.DNILider)
INNER JOIN Empleado_equipo eq ON (eq.codEquipo = Equipo.codEquipo)
WHERE descTecnologias = 'Java' AND codEquipo IN (
    SELECT codEquipo
```

```
    FROM Empleado_equipo
    GROUP BY codEquipo
    HAVING COUNT(codEquipo) = 1
)
```

6)

Consigna:

Modificar nombre, apellido y dirección del empleado con DNI 40568965 con los datos que desee.

Solución:

```
UPDATE Empleado SET nombre='Pipino' apellido='Cuevas' direccion = 'Cal
le 12 72'
WHERE DNI = 40568965
```

7)

Consigna:

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

Solución:

```
SELECT DNI, nombre, apellido, telefono, direccion
FROM Empleado
WHERE DNI IN (
    SELECT DNIResponsable
    FROM Proyecto
)
AND DNI NOT IN (
    SELECT DNILider
    FROM Equipo
)
```

8)

Consigna:

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

Solución:

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

9)

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

Consigna:

```
SELECT nombrP, descripcion, fechaInicioP, nombre, apellido
FROM Proyecto p INNER JOIN Empleado e ON (p.DNIResponsable = e.DNI)
WHERE fechaFinEstimada BETWEEN 1/1/2025 AND 31/12/2025
```

## Ejercicio 10

TABLAS:

**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)

Consigna:

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.

Solución:

```
SELECT patente, modelo, marca, peso, km
FROM Camion
WHERE cant_ruedas BETWEEN (4 AND 8)
AND patente IN (
   SELECT patente
   FROM Service
   WHERE feche BETWEEN 11/11/2023 AND 11/11/2023
)
```

2)

Consigna:

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”.

Solución:

```
SELECT patente, es_electrico, tipo_motor
FROM Auto a INNER JOIN Service s ON (s.patente = a.patente)
WHERE descripcion = 'cambio de aceite' AND km_service < 13000
UNION
SELECT patente, es_electrico, tipo_motor
FROM Auto a INNER JOIN Service s ON (s.patente = a.patente)
INNER JOIN Service_parte sp ON (sp.patente = s.patente AND sp.fecha =
s.fecha)
INNER JOIN Parte p ON (p.cod_parte = sp.cod_parte)
WHERE p.nombre = 'Filtro de combustible'
```

3)

Consigna:

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

Solución:

```
SELECT nombre, precio_parte
FROM Parte p INNER JOIN Service_Parte sp ON (p.cod_parte = sp.cod_par
te)
WHERE precio_parte > 4000
GROUP BY p.cod_parte, nombre, precio_parte
HAVING COUNT(p.cod_parte > 30)
```

4)

Consigna:

> Dar de baja todos los camiones con más de 250.000 km.

Solución:

```
DELETE FROM Service
WHERE patente IN (SELECT patente FROM Vehiculo v
INNER JOIN Camion c ON (c.patente = Camion.patente) WHERE v.km > 25
```

```
0000)
DELETE FROM Service_parte
WHERE patente IN (SELECT patente FROM Vehiculo v
INNER JOIN Camion c ON (c.patente = Camion.patente) WHERE v.km > 25
0000)
DELETE FROM Camion
WHERE patente IN (SELECT patente FROM Vehiculo v
INNER JOIN Camion c ON (c.patente = Camion.patente) WHERE v.km > 25
0000)
DELETE FROM Vehiculo
WHERE patente IN (SELECT patente FROM Vehiculo v WHERE v.km > 2500
00)
AND patente NOT IN (
SELECT patente FROM Vehiculo v
INNER JOIN Auto a ON  (c.patente = a.patente)
WHERE v.km > 250000
)
```

5)

Consigna:

> Listar el nombre y precio de aquellas partes que figuren en todos los service realizados en el año
> actual.

Solución:

```
SELECT nombre, precio
FROM Parte p
WHERE NOT EXISTS (
      SELECT *
      FROM Service s
      WHERE NOT EXISTS (
         SELECT *
         FROM Service_parte sp
         WHERE sp.cod_parte = p.cod_parte
         AND sp.fecha = s.fecha AND
         sp.patente = s.patente AND
```

```
            fecha BETWEEN 1/1/2024 AND 31/12/2024
        )
)
```

6)

Consigna:

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

Solución:

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

7)

Consigna:

> Dar de alta una parte, cuyo nombre sea “Aleron” y precio $5000.

Solución:

```
INSERT INTO Parte (nombre, precio) VALUES ('Aleron', 5000)
```

8)

Consigna:

> Dar de baja todos los services que se realizaron al auto con patente ‘AWA564’.

Solución:

```

```