Ejercicio 1

AGENCIA (​RAZON\_SOCIAL​, dirección, telef, e-mail)

CIUDAD (​CODIGOPOSTAL​, nombreCiudad, añoCreación)

CLIENTE (​DNI​, nombre, apellido, teléfono, dirección)

VIAJE​(FECHA,HORA,DNI​,cpOrigen(fk),cpDestino(fk),razon\_social(fk),descripción)

//cpOrigen y cpDestino corresponden a la ciudades origen y destino del viaje

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.

CREATE VIEW dniEspecifico(

SELECT dni

FROM cliente

WHERE (apellido=”Roma”))

CREATE VIEW codigoEspecifico(

SELECT codigopostal

FROM ciudad

WHERE (nombreCiudad=”La Plata”))

CREATE VIEW consulta(

SELECT DISTINCT(v.razon\_social)

FROM viaje v INNER JOIN dniEspecifico d ON (v.DNI=d.DNI)

INNER JOIN codigoEspecifico c ON (v.coOrigen=c.codigopostal))

SELECT a.razon\_social,a.direccion,a.telef

From consulta c INNER JOIN agencia a ON(c.razon\_social=a.razon\_social)

ORDERED BY telef,razon\_social

2. Listar fecha, hora, datos personales del cliente, ciudad origen y destino de viajes realizados en enero de 2019 donde la descripción del viaje contenga el String ‘demorado’.

CREATE VIEW viajeEspecifico(

SELECT \*

FROM viaje v

WHERE YEAR(v.fecha)=2019 AND v.descripcion LIKE “%demorado%”)

SELECT a.razon\_social,a.direccion,a.telef,c1.nombre,c2.nombre

From viajeEspecifico v INNER JOIN ciudad c1 ON(v.cpOrigen=c1.codigopostal)

INNER JOIN ciudad c2 ON(v.cpDestino=c2.codigopostal)

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

CREATE VIEW viajeEspecifico(

SELECT \*

FROM viaje v

WHERE YEAR(v.fecha)=2019)

CREATE VIEW agenciaEspecifica(

SELECT \*

FROM agencia a

WHERE YEAR(v.fecha)=2019)

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

SELECT DISTINCT(dni),nombre,apellido,telefono,direccion

FROM

(SELECT cli.dni,cli.nombre,cli.apellido,cli.telefono,cli.direccion

FROM cliente cli

INNER JOIN viaje v ON(cli.dni=v.dni)

INNER JOIN (SELECT \* FROM ciudad c WHERE c.nombre=”Coronel Brandsen”) ciu ON

(v.cpDestino=ciu.codigoPostal)

EXCEPT

(SELECT cli.dni,cli.nombre,cli.apellido,cli.telefono,cli.direccion

FROM cliente cli INNER JOIN viaje v ON(cli.dni=v.dni)

INNER JOIN (SELECT \* FROM ciudad c WHERE c.nombre<>”Coronel Brandsen”) ciu ON

(v.cpDestino=ciu.codigoPostal)))

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

SELECT COUNT(\*) as “cantidad de viajes de TAXI Y a Villa Elisa”

FROM agencia a

INNER JOIN viaje v ON (a.razon\_social=”TAXI Y” AND a.razon\_social=v.razon\_social)

INNER JOIN ciudad c ON (v.cpDestino=c.codigoPostal AND c.nombre=”Villa Elisa”)

GROUP BY a.razon\_social

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 EXIST (

SELECT \*

FROM agencia a

WHERE NOT EXIST (

SELECT \*

FROM viaje v

WHERE (v.razon\_social=a.razon\_social & v.dni=c.dni)))

7. Modificar el cliente con DNI: 38495444 actualizando el teléfono a: 221-4400897.

UPDATE cliente SET telefono WHERE dni=38495444

8. Listar razon\_social, dirección y teléfono de las agencias que tengan mayor cantidad de viajes realizados.

SELECT a.razon\_social,a.direccion,a.telefono

FROM (

SELECT a1.razon\_social,a1.direccion,a1.telefono,COUNT(\*) as cantidad

FROM agencia a1 INNER JOIN viaje v ON (v.razon\_social=a2.razon\_social)

GROUP BY a1.razon\_social,a1.direccion,a1.telefono)

HAVING COUNT(\*)=(

SELECT MAX(cantidad)

FROM (

SELECT COUNT(\*) as cantidad

FROM agencia a2 INNER JOIN viaje v ON (v.razon\_social=a2.razon\_social)

GROUP BY a.razon\_social)))

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

SELECT c.nombre,c.apellido,c.direccion,c.telefono

FROM (

SELECT c.nombre,c.apellido,c.direccion,c.telefono,COUNT(\*) as cantidad

FROM cliente c INNER JOIN viaje v ON (c.dni=v.dni)

GROUP BY c.dni,c.nombre,c.apellido,c.direccion,c.telefono

HAVING COUNT(\*) >=10)

10. Borrar al cliente con DNI 40325692.

DELETE FROM viaje WHERE dni=40325692

DELETE FROM cliente WHERE dni=40325692

Ejercicio 2

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

Factura (nroTicket, total, fecha, hora,idCliente (fk))

Detalle(nroTicket, idProducto, cantidad, preciounitario)

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

1. Listar datos personales de clientes cuyo apellido comience con el string ‘Pe’. Ordenar por DNI

SELECT \*

FROM cliente

WHERE (apellido LIKE “Pe%”)

ORDER BY dni

2. Listar nombre, apellido, DNI, teléfono y dirección de clientes que realizaron compras solamente durante 2017.

SELECT c.nombre,c.apellido,c.dni,c.telefono,c.direccion

FROM cliente c INNER JOIN factura f ON (c.idCliente=f.idCliente & YEAR(f.fecha)=2017)

EXCEPT (

SELECT c.nombre,c.apellido,c.dni,c.telefono,c.direccion

FROM cliente c INNER JOIN factura f ON (c.idCliente=f.idCliente & YEAR(f.fecha)<>2017))

3. Listar nombre, descripción, precio y stock de productos vendidos al cliente con DNI:45789456, pero que no fueron vendidos a clientes de apellido ‘Garcia’.

SELECT DISTINCT(p.nombreP as nombre,p.descripcion,p.precio,p.stock)

FROM producto p

INNER JOIN detalle d ON (p.idProducto=d.idProducto)

INNER JOIN factura f ON (d.nroTicket=f.nroTicket)

INNER JOIN cliente c ON (f.idCliente=c.idCliente & c.dni=45789456)

EXCEPT (

SELECT DISTINCT(p.nombreP as nombre,p.descripcion,p.precio,p.stock)

FROM producto p

INNER JOIN detalle d ON (p.idProducto=d.idProducto)

INNER JOIN factura f ON (d.nroTicket=f.nroTicket)

INNER JOIN cliente c ON (f.idCliente=c.idCliente & c.apellido=”Garcia”))

corregir lo de arriba y luego esto en consecuencia

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 DISTINCT(p.nombreP as nombre,p.descripcion,p.precio,p.stock)

FROM producto p

INNER JOIN detalle d ON (p.idProducto=d.idProducto)

INNER JOIN factura f ON (d.nroTicket=f.nroTicket)

INNER JOIN cliente c ON (f.idCliente=c.idCliente & c.telefono LIKE “221%”)

ORDER BY p.nombreP

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

SELECT p.nombreP as nombre,p.descripcion,p.precio,SUM(cantidad) as cantidad\_detalles

FROM producto p LEFT JOIN detalle d ON (p.idProducto=d.idProducto))

GROUP BY p.idProducto,p.nombreP as nombre,p.descripcion,p.precio

¿Cuando el SUM corra sobre las tuplas que tienen cantidad en null no se va a romper?

Usar un union y poner 0 en select

¿Está bien dejar en null los campos de cantidad en el LEFT JOIN o debemos hacer que queden con el valor cero?

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 DISTINCT(c.nombre,c.apellido,c.dni,c.telefono,c.direccion)

FROM cliente c

INNER JOIN factura f ON (c.idCliente=f.idCliente)

INNER JOIN detalle d ON (f.nroTicket=d.nroTicket)

INNER JOIN producto p ON (d.idProducto=p.idProducto & (p.nombreP=”prod1” | p.nombreP=”prod2”))

EXCEPT (

SELECT DISTINCT(c.nombre,c.apellido,c.dni,c.telefono,c.direccion)

FROM cliente c

INNER JOIN factura f ON (c.idCliente=f.idCliente)

INNER JOIN detalle d ON (f.nroTicket=d.nroTicket)

INNER JOIN producto p ON (d.idProducto=p.idProducto & p.nombreP=”prod3”))

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 2019.

SELECT DISTINCT(f.nroTicket,f.total,f.fecha,f.hora,f.idCliente)

FROM factura f

INNER JOIN detalle d ON (f.nroTicket=d.nroTicket & YEAR(f.fecha)=2019)

INNER JOIN producto p ON (d.idProducto=p.idProducto & p.nombreP=”prod38”)

8. 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.

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 DISTINCT(f.nroTicket,f.total,f.fecha,f.hora)

FROM cliente c

INNER JOIN factura f ON (c.idCliente=f.idCliente & c.cliente=”Jorje Perez”)

INNER JOIN detalle d ON (f.nroTicket=d.nroTicket)

INNER JOIN producto p ON (d.idProducto=p.idProducto & p.nombreP=”Z”)

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

SELECT DISTINCT(dni,apellido,nombre)

FROM (

SELECT c.dni,c.apellido,c.nombre,SUM(total) as monto\_total

FROM cliente c INNER JOIN factura f ON (c.idCliente=f.idCliente)

GROUP BY c.idCliente

HAVING monto\_total>10000000)

Ejercicio 3:

Club=(codigoClub, nombre, anioFundacion, codigoCiudad(FK))

Ciudad=(codigoCiudad, nombre)

Estadio=(codigoEstadio, codigoClub(FK), nombre, direccion)

Jugador=(DNI, nombre, apellido, edad, codigoCiudad(FK))

ClubJugador(codigoClub, DNI, desde, hasta)

1. Reportar nombre y anioFundacion de aquellos clubes de la ciudad de La Plata que no poseen estadio.

SELECT c.nombre,c.anioFundacion

FROM club c

WHERE NOT EXISTS (

SELECT \*

FROM estadio e

WHERE (c.codigoClub=e.codigoClub))

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 NOT EXISTS (

SELECT \*

FROM jugador j

INNER JOIN clubJugador cj ON (j.dni=cj.dni)

INNER JOIN ciudad ciu ON (ciu.codigoCiudad=j.codigoCiudad & ciu.nombre=”Berisso”)

WHERE (c.codigoClub=cj.codigoClub))

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

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)

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)

WHERE (j.edad>29) & EXISTS (

SELECT \*

FROM ciudad ciu

WHERE (ciu.nombre=”Córdoba” & ciu.codigoCiudad=c.codigoCiudad))

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 edad\_promedio

FROM club c

INNER JOIN clubJugador cj ON (j.codigoClub=cj.codigoClub)

INNER JOIN jugador j ON (cj.dni=j.dni)

WHERE (cj.hasta IS NOT 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(\*) as cantidad\_clubes

FROM jugador j INNER JOIN clubJugador cj ON (j.dni=cj.dni)

INNER JOIN club c ON (cj.codigoClub=c.codigoClub)

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 NOT EXISTS(

SELECT \*

FROM clubJugador cj

WHERE c.codigoClub=cj.codigoClub & EXISTS(

SELECT \*

FROM jugador j INNER JOIN ciudad ciu ON (j.codigoCiudad=ciu.codigoCiudad)

WHERE cj.dni=j.dni & ciu.nombre=”Mar del Plata”))

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

SELECT j.nombre,j.apellido

FROM jugador j

WHERE NOT EXISTS(

SELECT \*

FROM club c

WHERE NOT EXISTS(

SELECT \*

FROM clubJugador cj

WHERE j.dni=cj.dni & c.codigoClub=cj.codigoClub))

9. Agregar con codigoClub 1234 el club “Estrella de Berisso” 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.

INSERT INTO club (codigoClub,nombre,anioFundacion,codigoCiudad)

VALUES (1234,“Estrella de Berisso”,1921,(

SELECT codigo

FROM ciudad c

WHERE c.nombre=”Berisso”))

Ejercicio 4:

PERSONA = (DNI, Apellido, Nombre, Fecha\_Nacimiento, Estado\_Civil, Genero)

ALUMNO = (DNI, Legajo, Año\_Ingreso)

PROFESOR = (DNI, Matricula, Nro\_Expediente)

TITULO = (Cod\_Titulo, Nombre, Descripción)

TITULO-PROFESOR = (Cod\_Titulo, DNI, Fecha)

CURSO = (Cod\_Curso, Nombre, Descripción, Fecha\_Creacion, Duracion)

ALUMNO-CURSO = (DNI, Cod\_Curso, Año, Desempeño, Calificación)

PROFESOR-CURSO = (DNI, Cod\_Curso, Fecha\_Desde, Fecha\_Hasta)

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

SELECT a.dni,a.legajo,p.nombre

FROM alumno a INNER JOIN persona p ON (a.dni=p.dni)

WHERE (a.año\_ingreso=2014)

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

SELECT pf.dni,pf.matricula,p.nombre

FROM profesor pf INNER JOIN persona p ON (pf.dni=p.dni)

WHERE EXISTS (

SELECT \*

FROM profesor-curso pc INNER JOIN curso c ON (pc.cod\_curso=c.cod\_curso)

WHERE pf.dni=c.dni & c.duracion>100)

3. Listar el DNI, Apellido, Nombre, Género y Fecha de nacimiento de los alumnos inscriptos al curso con nombre “Diseño de Bases de Datos” en 2019.

SELECT DISTINCT (a.dni,p.apellido,p.nombre,p.genero,p.fecha\_nacimiento)

FROM alumno a

INNER JOIN persona p ON (a.dni=p.dni)

INNER JOIN alumno-curso ac ON (a.dni=ac.dni)

INNER JOIN curso c ON (ac.cod\_curso=c.cod\_curso)

WHERE ac.año=2019 & c.nombre=”DISEÑO de Bases de Datos”

Nota: Agrege el distinct por que nada me asegura que solo haya un curso que se llame “Diseño de Bases de Datos”

4. Listar el DNI, Apellido, Nombre y Calificación de aquellos alumnos que obtuvieron una calificación superior a 9 en los cursos que dicta el profesor “Juan Garcia”. Dicho listado deberá estar ordenado por Apellido.

SELECT a.dni,p1.apellido,p1.nombre,c.nombre,ac.calificacion

FROM alumno a

INNER JOIN persona p1 ON (a.dni=p.dni)

INNER JOIN alumno-curso ac ON (a.dni=ac.dni)

INNER JOIN curso c ON (ac.cod\_curso=c.cod\_curso)

INNER JOIN profesor-curso pc ON (c.cod\_curso=pc.cod\_curso)

INNER JOIN profesor pf ON (pc.dni=pf.dni)

INNER JOIN persona p2 ON (pf.dni=p2.dni)

WHERE ac.calificacion>9 & p2.nombre=”Juan Garcia”

ORDER BY p1.apellido

Nota: Agregue el campo nombre de curso en el informe para saber de qué curso es la calificación que estoy informando

5. 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.

SELECT pf.dni,p.apellido,p.nombre,pf.matricula

FROM profesor pf

INNER JOIN persona p ON (pf.dni=p.dni)

INNER JOIN titulo-profesor tp ON (pf.dni=tp.dni)

INNER JOIN titulo t ON (to.cod\_titulo=t.cod\_titulo)

GROUP BY pf.dni,p.apellido,p.nombre,pf.matricula

HAVING COUNT(\*)>3

ORDER BY p.apellido,p.nombre

corregir lo de arriba y luego esto en consecuencia (mirar uso del COUNT(\*))

6. 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.

SELECT pf.dni,p.apellido,p.nombre,SUM(c.duracion) as cantidad\_horas,AVG(c.duracion) as promedio

FROM profesor pf

INNER JOIN persona p ON (pf.dni=p.dni)

INNER JOIN profesor-curso pc ON (pc.dni=pf.dni)

INNER JOIN curso c ON (pc.cod\_curso=c.cod\_curso)

GROUP BY pf.dni,p.apellido,p.nombre

7. Listar Nombre, Descripción del curso que posea más alumnos inscriptos y del que posea menos alumnos inscriptos durante 2019.

SELECT c.nombre,c.descripcion,MAX(cantidad\_inscriptos)

FROM (

SELECT c.nombre,c.descripcion,COUNT(\*) as cantidad\_inscriptos

FROM curso c

INNER JOIN alumno-curso ac ON (c.dni=ac.dni)

INNER JOIN alumno a ON (ac.cod\_curso=a.cod\_curso)

GROUP BY c.cod\_curso,c.nombre,c.descripcion)

SELECT c.nombre,c.descripcion,MIN(cantidad\_inscriptos)

FROM (

SELECT c.nombre,c.descripcion,COUNT(\*) as cantidad\_inscriptos

FROM curso c

INNER JOIN alumno-curso ac ON (ac.cod\_curso=c.cod\_curso)

INNER JOIN alumno a ON (a.dni=ac.dni)

GROUP BY c.cod\_curso,c.nombre,c.descripcion)

8. Listar el DNI, Apellido, Nombre, Legajo de alumnos que realizaron cursos con nombre conteniendo el string ‘BD’ durante 2018 pero no realizaron ningún curso durante 2019.

SELECT a.dni,p.apellido,p.nombre,a.legajo

FROM alumno a

INNER JOIN persona p ON (pf.dni=p.dni)

INNER JOIN alumno-curso ac ON (ac.dni=a.dni)

INNER JOIN curso c ON (c.cod\_curso=ac.cod\_curso)

WHERE c.nombre LIKE “%BD%” & ac.año=2018

EXCEPT (

SELECT a.dni,p.apellido,p.nombre,a.legajo

FROM alumno a

INNER JOIN persona p ON (pf.dni=p.dni)

INNER JOIN alumno-curso ac ON (ac.dni=a.dni)

INNER JOIN curso c ON (c.cod\_curso=ac.cod\_curso)

WHERE ac.año<>2018)

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

INSERT INTO persona (dni,apellido,nombre,fecha\_nacimiento,estado\_civil,genero) 1sansant

VALUES (22222222,”Smith”,”Jonh”,”22/7/1987”,casado,masculino)

INSERT INTO profesor (dni,matricula,nro\_expediente) VALUES (22222222,12345/6,2271987)

INSERT INTO título-profesor (cod\_titulo,dni,fecha) VALUES (25,22222222,”22/3/2020”)

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

UPDATE persona p SET estado\_civil=”divorciado” WHERE p.dni “dni del alumno con legajo 2020/09”

Corregir ese y luego venir aquí.

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

DELETE FROM persona p WHERE p.dni=30568989

DELETE FROM alumno a WHERE a.dni=30568989

DELETE FROM alumno-curso ac WHERE a.dni=30568989

Ejercicio 5:

Localidad(CodigoPostal, nombreL, descripcion, #habitantes) ¿Que significa el #?

Arbol(nroArbol, especie, años, 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’.

2. Reportar DNI, nombre, apellido, fnac y localidad donde viven podadores que tengan podas durante 2018.

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

4. Reportar especie, años,calle, nro y localidad de árboles que fueron podados durante 2017 y no fueron podados durante 2018.

5. Reportar DNI, nombre, apellido, fnac y localidad donde viven podadores con apellido terminado con el string ‘ata’ y que el podador tenga al menos una poda durante 2018. Ordenar por apellido y nombre.

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

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

8. Eliminar el podador con DNI: 22234566.

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