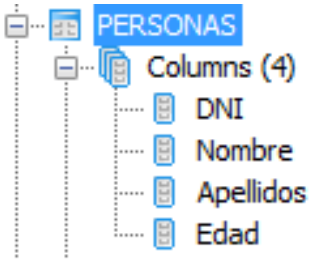
MANIPULAR BD POSTGRESQL

1- DEFINE las siguientes tablas a través de la interfaz gráfica en Windows e INSERTA datos en ella

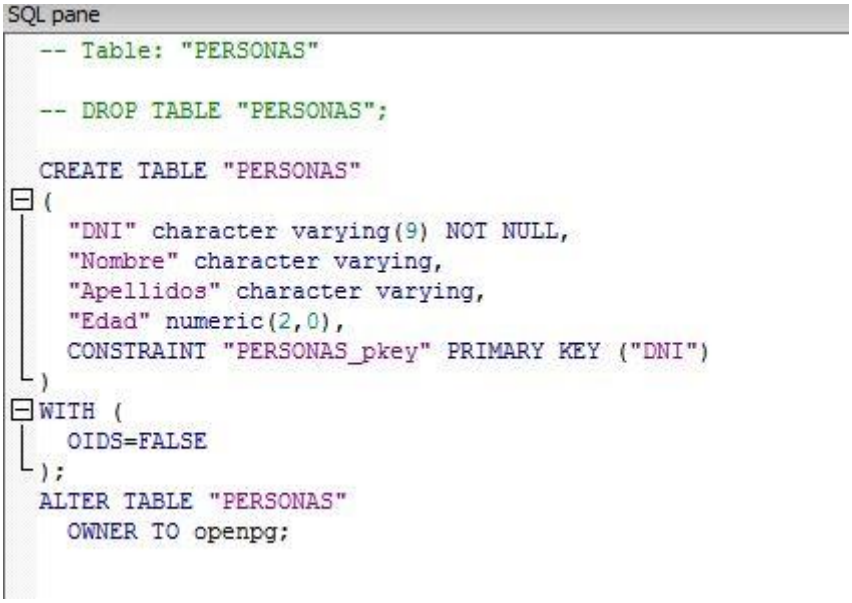
1.1 Previamente necesitarás conocer los tipos de datos que tiene Psql. Disponibles en el siguiente enlace: http://www.postgresql.org.ar/trac/wiki/datatype.html

1.2 Crear Tabla PERSONAS:

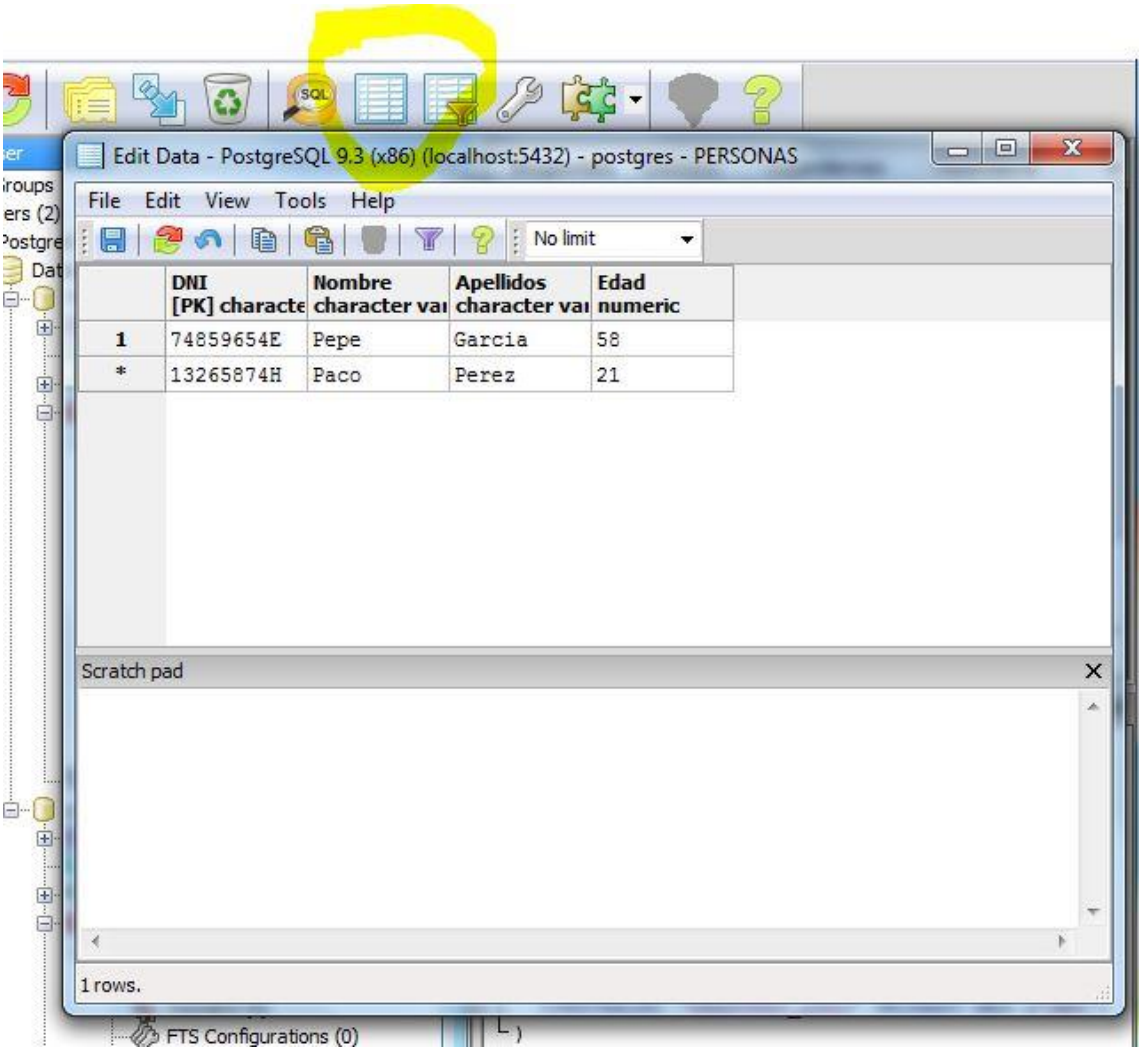
- DNI: Debe incluir 9 caracteres alfanuméricos. Será PK
- Nombre: caracteres
- Apellidos: caracteres
- Edad: entero menor de 99



1.3 Pega aquí el pantallazo de la/s sentencia/s SQL resultante/s de la configuración que has hecho gráficamente.



1.4 Busca dónde introducir 2 filas de datos y pega aquí el pantallazo



2- LEE los siguientes comandos, RECUERDA o BUSCA otros comandos SQL que vayas a necesitar, CREA las tablas por la terminal en Ubuntu e INSERTA datos

2.1 COMANDOS:

1) Comando: **\h**

Este comando sirve para ver la ayuda con respecto a la sintaxis de nuestras consultas SQL, por ejemplo: \h INSERT

```
Available help:
  ABORT                            DEALLOCATE
  ALTER AGGREGATE                  DECLARE
  ALTER COLLATION                  DELETE
  ALTER CONVERSION                 DISCARD
  ALTER DATABASE                   DO
  ALTER DEFAULT PRIVILEGES         DROP AGGREGATE
  ALTER DOMAIN                     DROP CAST
  ALTER EVENT TRIGGER              DROP COLLATION
```

Al utilizar este comando nos mostrará cómo debemos escribir la sentencia “insert”, así como parámetros y el orden correspondiente:

```
postgres=# \h INSERT
Command:     INSERT
Description: create new rows in a table
Syntax:
[ WITH [ RECURSIVE ] with_query [, ...] ]
INSERT INTO table_name [ ( column_name [, ...] ) ]
    { DEFAULT VALUES | VALUES ( { expression | DEFAULT } [, ...] ) [, ...] | que
ry }
    [ RETURNING * | output_expression [ [ AS ] output_name ] [, ...] ]

postgres=#
```

2) Comando: **\dt**

Mostrará la lista de las tablas de la base de datos que tengamos seleccionada.

```
postgres=# \dt
No relations found.
```

3) Comando: **\?**

Este comando nos mostrará una lista de todos los comandos que podemos usar en la consola interactiva de postgresql.

Para desplazarnos por la lista ocupamos las flechas de nuestro teclado y para salir solo presionamos “q”.

```
General
  \copyright            show PostgreSQL usage and distribution terms
  \g [FILE] or ;        execute query (and send results to file or |pipe)
  \gset [PREFIX]        execute query and store results in psql variables
  \h [NAME]             help on syntax of SQL commands, * for all commands
  \q                    quit psql
  \watch [SEC]          execute query every SEC seconds

Query Buffer
  \e [FILE] [LINE]      edit the query buffer (or file) with external editor
  \ef [FUNCNAME [LINE]] edit function definition with external editor
  \p                    show the contents of the query buffer
  \r                    reset (clear) the query buffer
  \s [FILE]             display history or save it to file
  \w FILE               write query buffer to file

Input/Output
  \copy ...             perform SQL COPY with data stream to the client host
  \echo [STRING]        write string to standard output
  \i FILE               execute commands from file
  \ir FILE              as \i, but relative to location of current script
  \o [FILE]             send all query results to file or |pipe
  \qecho [STRING]       write string to query output stream (see \o)

Informational
  (options: S = show system objects, + = additional detail)
  \d[S+]                list tables, views, and sequences
  \d[S+]  NAME          describe table, view, sequence, or index
  \da[S]  [PATTERN]     list aggregates
  \db[+]  [PATTERN]     list tablespaces
  \dc[S+] [PATTERN]     list conversions
  \dC[+]  [PATTERN]     list casts
  \dd[S]  [PATTERN]     show object descriptions not displayed elsewhere
  \ddp    [PATTERN]     list default privileges
  \dD[S+] [PATTERN]     list domains
:
```

4) Comando: **\conninfo**

Este comando nos mostrará la información de nuestra conexión activa.

```
postgres=# \conninfo
You are connected to database "postgres" as user "postgres" via socket in "/var/run/postgresq
l" at port "5432".
postgres=#
```

5) Iniciar la consola: **psql -U** *user* **-W -h** *host database*

6) Comando: \l

Lista las bases de datos

```
                                  List of databases
   Name    |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileg
es
-----------+----------+----------+-------------+-------------+------------------
-----
 ERP       | postgres | UTF8     | es_ES.UTF-8 | es_ES.UTF-8 |
 postgres  | postgres | UTF8     | es_ES.UTF-8 | es_ES.UTF-8 |
 template0 | postgres | UTF8     | es_ES.UTF-8 | es_ES.UTF-8 | =c/postgres
    +
           |          |          |             |             | postgres=CTc/post
gres
 template1 | postgres | UTF8     | es_ES.UTF-8 | es_ES.UTF-8 | =c/postgres
    +
           |          |          |             |             | postgres=CTc/post
gres
(4 rows)

~
~
~
~
~
~
(END)
```

7) Comando: **\c** *DB*
Seleccionar una base de datos o cambiar de BD

```
postgres=# \c ERP
You are now connected to database "ERP" as user "postgres".
```

8) Comando: **\d** *TABLA*
Para ver la información de la estructura de una tabla (=DESC)

```
ERP=# \d
                List of relations
 Schema |         Name          |   Type   |  Owner
--------+-----------------------+----------+----------
 public | PruebaTabla1          | table    | postgres
 public | PruebaTabla1_ID_seq   | sequence | postgres
(2 rows)

ERP=# \d 'PruebaTabla1'
Did not find any relation named "PruebaTabla1".
```

9) Vaciar una tabla:
TRUNCATE TABLE *table* **RESTART IDENTITY**
Con este comando borramos el contenido de una tabla y reiniciamos su índice sino agregamos RESTART IDENTITY nuestros índices no serán reiniciados y seguirán según el último registro

```
ERP=# TRUNCATE TABLE PruebaTabla1 RESTART IDENTITY
```

10) Crear una base de datos:
CREATE DATABASE *basename*;

```
ERP=# CREATE DATABASE nuevaBBDD;
CREATE DATABASE
ERP=#
```

11) Borrar o eliminar una base de datos:
DROP DATABASE *basename*;

```
nuevabbdd=# DROP TABLE pruebatabla;
DROP TABLE
```

12) Borrar o eliminar una tabla:
DROP TABLE *tablename*;

```
nuevabbdd=# DROP TABLE pruebatabla;
DROP TABLE
```

13) Uso de comillas:
SELECT “column” FROM “table” WHERE “column” = 'value';

Generalmente utilizar comillas dobles (“) para columnas y comillas simples (') para valores. No es una regla pero es necesario cuando utilizamos nombres reservados, por ejemplo:

```
postgres=# SELECT * from pruebatabla;
 nombre | apellido
--------+----------
(0 rows)
```

SELECT **to** FROM table;
En este caso tenemos un campo llamado “to”, esto nos dará un error de sintaxis, por lo tanto tendremos que usar comillas dobles:

```
postgres=# SELECT "nombre" from pruebatabla;
 nombre
--------
(0 rows)
```

SELECT **“to”** FROM table;

14) Comando:**\q**
Salir del cliente psql:

```
postgres=# \q
postgres@borja-VirtualBox:/home/borja$
```

2.2 Crea la siguiente tabla:
Tabla: ANIMALES

- Id: campo entero que se autoincrementa. Será PK
- Nombre: caracteres. Longitud 10
- Fecha_Nacimiento. Fecha
- Dueño: caracteres. Longitud 30
- Dirección. Caracteres. Longitud 30

```
postgres=# CREATE TABLE ANIMALES (id  SERIAL  PRIMARY KEY, nombre char(10), Fech
a_nacimiento date, dueño char(30), direccion char(30));
CREATE TABLE
postgres=# \dt
            List of relations
 Schema |    Name     | Type  |  Owner
--------+-------------+-------+----------
 public | animales    | table | postgres
 public | pruebatabla | table | postgres
(2 rows)
```

2.3 Introduce 2 filas de datos.

```
postgres=# insert into animales values(1,'Perro', '10/10/2000','Pepe','C\Alava' );
INSERT 0 1
postgres=# insert into animales values(2,'Borja', '11/12/2005','Paco','C\Dato' );
INSERT 0 1
```

2.4 Saca la descripción de la tabla para ver de qué tipo son sus columnas

```
postgres=# \d animales
                              Table "public.animales"
      Column      |     Type      |                       Modifiers
------------------+---------------+-------------------------------------------------------
 id               | integer       | not null default nextval('animales_id_seq'::regclass)
 nombre           | character(10) |
 fecha_nacimiento | date          |
 dueño            | character(30) |
 direccion        | character(30) |
Indexes:
    "animales_pkey" PRIMARY KEY, btree (id)
```

2.5 Lista los valores introducidos en la tabla.

```
 id |   nombre   | fecha_nacimiento |             dueño              |           direccion
----+------------+------------------+--------------------------------+--------------------------
------
  1 | Perro      | 2000-10-10       | Pepe                           | C\Alava

  2 | Borja      | 2005-12-11       | Paco                           | C\Dato

(2 rows)
```