

PostgreSQL

Instalación Windows

Trabajando desde consola

Parámetros de conexión

Acceder a Psql

Prompts de consola

Comandos Psql

Demostración comandos

\l

\c

\d

\ds

\di

\dp \z

Creación de bases de datos

Tipos de datos

Tipos de Datos Numéricos

Tipos de Datos de Caracteres

Tipos de Datos Booleanos

Tipos de Datos de Fecha y Hora(<https://www.postgresql.org/docs/current/datatype-datetime.html>)

Tipos de Datos Monetarios

Tipos de Datos Binarios (<https://www.postgresql.org/docs/current/datatype-binary.html>)

Tipos de Datos de Redes (<https://www.postgresql.org/docs/current/datatype-net-types.html>)

Tipos de Datos Geométricos (<https://www.postgresql.org/docs/current/datatype-geometric.html>)

Tipos de Datos JSON y XML

Tipos de Datos Especiales

Tipos de Datos Enumerados

Ejemplo General

Ejemplo de Inserción de Datos

Creación de tablas

Constraints (Restricciones)

Tipos de Constraints

Explicación de los Constraints

Añadir Constraints a una Tabla Existente

Añadir PRIMARY KEY

Añadir FOREIGN KEY

Añadir UNIQUE

Añadir NOT NULL

Añadir CHECK

Añadir DEFAULT

Ejercicio Tablas y Constraints

Volcado de datos

Fecha, DateTime e Intervalos

`CURRENT_DATE`

`CURRENT_TIME`

`CURRENT_TIMESTAMP`

`now()`

`AGE()`

`EXTRACT()`

PostgreSQL

PostgreSQL, que significa "Postgres Structured Query Language", fue desarrollado en la Universidad de California, Berkeley, en la década de 1980. El ilustre científico de la computación Michael Stonebraker lideró un equipo de investigadores que construyó un sistema de gestión de

bases de datos relacional llamado Ingres. Más tarde, algunos de los miembros de ese equipo, incluido Stonebraker, se dispersaron y formaron el proyecto Postgres para continuar innovando en el área de la tecnología de bases de datos.

Instalación Windows

Ver video : <https://youtu.be/6U1M8TdCdgs?si=48v23fpifwS6WwRW>

Trabajando desde consola

Parámetros de conexión

```
Opciones de conexión:
-h, --host=NOMBRE      nombre del anfitrión o directorio de socket
                        (por omisión: «socket local»)
-p, --port=PUERTO      puerto del servidor (por omisión: «5432»)
-U, --username=NOMBRE  nombre de usuario (por omisión: «developer»)
-w, --no-password      nunca pedir contraseña
-W, --password         forzar petición de contraseña
                        (debería ser automático)
```

Acceder a Psql

```
psql --host=localhost --username=postgres --password=1234
```

o

```
psql --host=localhost -U postgres
```

[Ir a Creacion de bases de datos](#)

```
Reporte de errores a <pgsql-bugs@lists.postgresql.org>.
Sitio web de PostgreSQL: <https://www.postgresql.org/>
PS C:\Users\developer> psql --host=localhost --username=postgres --password=1234
Contraseña:
psql (15.6)
ADVERTENCIA: El código de página de la consola (850) difiere del código
de página de Windows (1252).
Los caracteres de 8 bits pueden funcionar incorrectamente.
Vea la página de referencia de psql «Notes for Windows users»
para obtener más detalles.
Digite «help» para obtener ayuda.
postgres=#
```

Ingrese la contraseña del usuario administrador (postgres) o usuario de preferencia

Prompts de consola

Indicador	Significado
=#	Espera una nueva sentencia. En psql, =# es el prompt que indica que uno está conectado de manera interactiva a la base de datos PostgreSQL. Cada vez que se ve =# en la terminal, significa que se puede introducir una sentencia SQL o un comando interactivo de psql para comunicarse con la base de datos.
-#	La sentencia aún no se ha terminado con ";" o \g
"#	Una cadena en comillas dobles no se ha cerrado
'#	Una cadena en comillas simples no se ha cerrado

(#	Un paréntesis no se ha cerrado
\?	Muestra todas las variantes de comandos, q para terminar

Comandos Psql

Comando	Descripción
\l	Lista las bases de datos
\d	Describe las tablas de la base de datos en uso
\ds	Lista las secuencias, estas se crean automáticamente cuando se declaran columnas de tipo serial.
\di	Lista los índices
\dv	Lista las vistas
\dp \z	Lista los privilegios sobre las tablas
\da	Lista las funciones de agregados
\df	Lista las funciones
\g archivo	Ejecuta los comandos de archivo
\c	Seleccionar base de datos:

Demostración comandos

\l

Nombre	Dueño	Codificación	Collate
dbfarm	postgres	UTF8	Spanish_Colombia.1252
demobusqueda	postgres	UTF8	Spanish_Colombia.1252
electrowarehouse	postgres	UTF8	Spanish_Colombia.1252

Nombre	Dueño	Codificación	Collate
geofarm	postgres	UTF8	Spanish_Colombia.1252
introj2	postgres	UTF8	Spanish_Colombia.1252
postgres	postgres	UTF8	Spanish_Colombia.1252
template0	postgres	UTF8	Spanish_Colombia.1252
template1	postgres	UTF8	Spanish_Colombia.1252

\c

Cuando se realiza seleccion de una base de datos diferente postgres solicita nuevamente la contraseña de administrador.

```
postgres=# \c geografica
Contraseña:
```

Ahora está conectado a la base de datos «geografica» con el usuario «postgres». postgres=#

\d

```
geografica=# \d
```

Esquema	Nombre	Tipo	Dueño
public	addresscustomer	tabla	postgres
public	addresscustomer_id_seq	secuencia	postgres
public	addressdispensary	tabla	postgres
public	addressdispensary_id_seq	secuencia	postgres
public	addressfarmacy	tabla	postgres
public	addressfarmacy_id_seq	secuencia	postgres
public	adminmode	tabla	postgres

\ds

Esquema	Nombre	Tipo	Dueño
public	addresscustomer_id_seq	secuencia	postgres
public	addressdispensary_id_seq	secuencia	postgres
public	addressfarmacy_id_seq	secuencia	postgres
public	adminmode_id_seq	secuencia	postgres

Esquema	Nombre	Tipo	Dueño
public	conveyor_idconve_seq	secuencia	postgres
public	laboratory_id_seq	secuencia	postgres
public	location_id_seq	secuencia	postgres

\di

Esquema	Nombre	Tipo	Dueño	Tabla
public	addresscustomer_pk	Índice	postgres	addresscustomer
public	addressdispensary_pk	Índice	postgres	addressdispensary
public	addressfarmacy_pk	Índice	postgres	addressfarmacy
public	adminmode_pk	Índice	postgres	adminmode
public	adminmode_unique	Índice	postgres	adminmode
public	city_pk	Índice	postgres	city
public	city_unique	Índice	postgres	city

\dp \z

Esquema	Nombre	Tipo	Privilegios	Privilegios de acceso a columnas	Políticas
public	addresscustomer	tabla			
public	addresscustomer_id_seq	secuencia			
public	addressdispensary	tabla			
public	addressdispensary_id_seq	secuencia			
public	addressfarmacy	tabla			

Creación de bases de datos

1. Salir de la base datos seleccionada en caso de tener una seleccionada. Use el comando \q
2. Inicie sesión nuevamente [Acceder a Psql](#)
3. Ingrese el comando DDL para la creación de la base de datos de ejemplo (college)

```
postgres=# CREATE DATABASE college; CREATE
DATABASE
postgres=#
```

4. Seleccione la base de datos creada \c [Nombre DB]

```
postgres=# \c college;
Ahora está conectado a la base de datos «college» con el usuario «postgres». college=#
```

Tipos de datos

Tipos de Datos Numéricos

Name	Storage Size	Description	Range
<code>smallint</code>	2 bytes	small-range integer	-32768 to +32767
<code>integer</code>	4 bytes	typical choice for integer	-2147483648 to +2147483647
<code>bigint</code>	8 bytes	large-range integer	-9223372036854775808 to +9223372036854775807
<code>decimal</code>	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
<code>numeric</code>	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
<code>real</code>	4 bytes	variable-precision, inexact	6 decimal digits precision
<code>double precision</code>	8 bytes	variable-precision, inexact	15 decimal digits precision
<code>smallserial</code>	2 bytes	small autoincrementing integer	1 to 32767
<code>serial</code>	4 bytes	autoincrementing integer	1 to 2147483647
<code>bigserial</code>	8 bytes	large autoincrementing integer	1 to 9223372036854775807

Tipos de Datos de Caracteres

Name	Description
<code>character varying(* n *)</code> , <code>varchar(* n *)</code>	variable-length with limit
<code>character(* n *)</code> , <code>char(* n *)</code> , <code>bpchar(* n *)</code>	fixed-length, blank-padded
<code>bpchar</code>	variable unlimited length, blank-trimmed

Name	Description
<code>text</code>	variable unlimited length

Tipos de Datos Booleanos

Name	Storage Size	Description
<code>boolean</code>	1 byte	state of true or false

Tipos de Datos de Fecha y Hora(<https://www.postgresql.org/docs/current/datatype-datetime.html>)

Name	Storage Size	Description	Low Value	High Value	Resolution
<code>timestamp [(*p*)] [without time zone]</code>	8 bytes	both date and time (no time zone)	4713 BC	294276 AD	1 microsecond
<code>timestamp [(*p*)] with time zone</code>	8 bytes	both date and time, with time zone	4713 BC	294276 AD	1 microsecond
<code>date</code>	4 bytes	date (no time of day)	4713 BC	5874897 AD	1 day
<code>time [(*p*)] [without time zone]</code>	8 bytes	time of day (no date)	00:00:00	24:00:00	1 microsecond
<code>time [(*p*)] with time zone</code>	12 bytes	time of day (no date), with time zone	00:00:00+1559	24:00:00-1559	1 microsecond
<code>interval [fields] [(*p*)]</code>	16 bytes	time interval	-178000000 years	178000000 years	1 microsecond

Tipos de Datos Monetarios

Name	Storage Size	Description	Range
<code>money</code>	8 bytes	currency amount	-92233720368547758.08 to +92233720368547758.07

Tipos de Datos Binarios (<https://www.postgresql.org/docs/current/datatype-binary.html>)

Una cadena binaria es una secuencia de octetos (o bytes). Las cadenas binarias se distinguen de las cadenas de caracteres de dos maneras. Primero, las cadenas binarias permiten específicamente almacenar octetos con valor cero y otros octetos "no imprimibles" (generalmente, octetos fuera del rango decimal de 32 a 126). Las cadenas de caracteres no permiten octetos con valor cero, ni tampoco permiten cualquier otro valor de octeto y secuencias de valores de octetos que sean inválidos según la codificación del conjunto de caracteres seleccionados en la base de datos. En segundo lugar, las operaciones en cadenas binarias procesan los bytes reales, mientras que el procesamiento de cadenas de caracteres depende de la configuración regional. En resumen, las cadenas binarias son apropiadas para almacenar datos que el programador considera como "bytes en bruto", mientras que las cadenas de caracteres son apropiadas para almacenar texto.

El tipo `bytea` soporta dos formatos para entrada y salida: el formato "hex" y el formato "escape" histórico de PostgreSQL. Ambos formatos son siempre aceptados en la entrada. El formato de salida depende del parámetro de configuración `bytea_output`; el valor predeterminado es hex. (Tenga en cuenta que el formato hex se introdujo en PostgreSQL 9.0; las versiones anteriores y algunas herramientas no lo entienden).

Name	Storage Size	Description
<code>bytea</code>	1 or 4 bytes plus the actual binary string	variable-length binary string

Tipos de Datos de Redes (<https://www.postgresql.org/docs/current/datatype-net-types.html>)

Name	Storage Size	Description
<code>cidr</code>	7 or 19 bytes	IPv4 and IPv6 networks
<code>inet</code>	7 or 19 bytes	IPv4 and IPv6 hosts and networks
<code>macaddr</code>	6 bytes	MAC addresses
<code>macaddr8</code>	8 bytes	MAC addresses (EUI-64 format)

Tipos de Datos Geométricos (<https://www.postgresql.org/docs/current/datatype-geometric.html>)

Name	Storage Size	Description	Representation
<code>point</code>	16 bytes	Point on a plane	(x,y)
<code>line</code>	32 bytes	Infinite line	{A,B,C}

<code>line</code>	32 bytes	Finite line segment	((x1,y1),(x2,y2))
<code>box</code>	32 bytes	Rectangular box	((x1,y1),(x2,y2))
<code>path</code>	16+16n bytes	Closed path (similar to polygon)	((x1,y1),...)
<code>open</code>	16+16n bytes	Open path	[(x1,y1),...]
<code>polygon</code>	40+16n bytes	Polygon (similar to closed path)	((x1,y1),...)
<code>circle</code>	24 bytes	Circle	<(x,y),r> (center point and radius)

Tipos de Datos JSON y XML

`json`: Datos en formato JSON

`jsonb`: Datos JSON en un formato binario más eficiente

`xml`: Datos en formato XML

Tipos de Datos Especiales

`uuid`: Identificador único universal

`array`: Arreglos de cualquier tipo de datos

`composite`: Tipo compuesto de varios tipos de datos

`range`: Rango de valores

Tipos de Datos Enumerados

Los tipos enumerados se crean utilizando el comando `CREATE TYPE`, por ejemplo:

`enum`: Conjunto de valores predefinidos

```
CREATE TYPE mood AS ENUM ('sad', 'ok', 'happy');
CREATE TYPE estatus AS ENUM ('casado(a)', 'soltero(a)', 'viudo(a)', 'con pelo(a)');
CREATE TABLE person (
    name text,
    current_mood mood
);
INSERT INTO person VALUES ('Moe', 'happy');
SELECT * FROM person WHERE current_mood = 'happy';
name | current_mood
-----+-----
Moe  | happy
```

Ejemplo General

```
CREATE TABLE ejemplo (
    id serial PRIMARY KEY, -- serial: Entero con auto-incremento nombre varchar(100) NOT NULL, -- varchar: Cadena de longitud variable con limite
    descripcion text, -- text: Cadena de longitud variable sin limite
    precio numeric(10, 2) NOT NULL, -- numeric: Numero de precision arbitraria
    on_stock boolean NOT NULL, -- boolean: Valor booleano fecha_creacion date NOT NULL, -- date: Fecha (sin hora) hora_creacion time NOT NULL, -- time:
    hora del dia (sin fecha) fecha_hora timestamp NOT NULL, -- timestamp: Fecha y hora (sin fuso horario)
    fecha_hora_zone timestamp with time zone, -- timestamp with time zone: Fecha y hora con fuso horario
    duracion interval, -- interval: Periodo de tiempo direccion_ip inet, -- inet: Direccion IP
    direccion_mre macaddr, -- macaddr: Direccion MAC punto_geometrico point, -- point: Punto en un plano de dos dimensiones
    datos_json json, -- json: Datos en formato JSON datos_jsonb jsonb, -- jsonb: Datos JSON en un formato binario mas eficiente
    identificador_unico uuid, -- uuid: Identificador unico universal
    estado_monetario money, -- money: Cantidad monetaria rangos int4range, -- range: Rango de valores colores_preferidos varchar(20)[] -- array: Arreglo
    de cadenas );
```

-- Ejemplo de tipo ENUM

```
CREATE TYPE estado_civil AS ENUM ('Casado(a)', 'Soltero(a)', 'Viudo(a)', 'Con pelo(a)');
CREATE TYPE estado_pedido AS ENUM ('pendiente', 'en_proceso', 'completado', 'cancelado');
```

```
CREATE TABLE pedidos (
    id serial PRIMARY KEY,
    estado estado_pedido NOT NULL -- enum: Conjunto de valores predefinidos
);
```

-- Ejemplo de tipo compuesto

```
CREATE TYPE direccion_completa AS (
    calle varchar,
    ciudad varchar,
    codigo_postal integer
);
```

```
CREATE TABLE clientes (
    id serial PRIMARY KEY,
    nombre varchar(50),
    ocivil estado_pedido,
    direccion direccion_completa -- compuesto: Tipo compuesto );
```

Ejemplo de Inserción de Datos

-- Insertando datos en la tabla 'ejemplo'

```
INSERT INTO ejemplo (
    nombre, descripcion, precio, on_stock, fecha_creacion, hora_creacion, fecha_hora, fecha_hora_zone, duracion, direccion_ip, direccion_mre,
    punto_geometrico, datos_json, datos_jsonb, identificador_unico, estado_monetario, rangos, colores_preferidos
) VALUES (
    'Producto A', 'Descripción del producto A', 29.99, true, '2024-07-09', '14:30:00',
    '2024-07-09 14:30:00', '2024-07-09 14:30:00+00', '1 day', '192.168.1.1', '08:00:27:00:00:00',
    '(10, 20)', '{"key": "value"}', '{"key": "value"}', '550e8400-e29b-41d4-a716-446655440000',
    'USD 100.00', '[10, 20]', ARRAY['rojo', 'verde', 'azul']
);
```

-- Insertando datos en la tabla 'pedidos'

```
INSERT INTO pedidos (estado) VALUES ('pendiente');
```

```
-- Insertando datos en la tabla 'clientes'
INSERT INTO clientes (direccion) VALUES (ROW('Calle Falsa 123', 'Springfield', 12345));
```

Creación de tablas

Para la creación de tablas se usa el comando CREATE TABLE

Sintaxis

```
CREATE TABLE nombre_tabla (
    columna1 tipo_dato [CONSTRAINT restricciones],
    columna2 tipo_dato [CONSTRAINT restricciones],
    ...
    [CONSTRAINT nombre_restriccion restriccion (columna1, columna2, ...)] );
```

Ejemplo

```
CREATE TABLE empleados (
    id serial PRIMARY KEY,
    nombre varchar(100) NOT NULL,
    edad integer NOT NULL,
    salario numeric(10, 2),
    fecha_contratacion date,
    active boolean DEFAULT true
);
```

En psql

```
colago=# CREATE TABLE empleados (
colago=# id serial PRIMARY KEY,
colago=# nombre varchar(100) NOT NULL,
colago=# edad integer NOT NULL,
colago=# salario numeric(10, 2),
colago=# fecha_contratacion date,
colago=# active boolean DEFAULT true
colago=# );
colago=# CREATE TABLE
```

Ejercicio:

Valide la creación de la tabla (\dt)

Esquema	Nombre	Tipo	Dueño
public	empleados	tabla	postgres

Verifique la estructura la tabla creada (\d empleados)

Columna	Tipo	Ordenamiento	Nulable	Por omisión
id	integer		not null	nextval('empleados_id_seq'::regclass)

nombre	character varying(100)		not null	
edad	integer		not null	
salario	numeric(10,2)			
fecha_contratacion	date			
activo	boolean			true

Constraints (Restricciones)

En PostgreSQL, los constraints (restricciones) se utilizan para definir reglas sobre los datos que se pueden almacenar en una tabla. Aquí te muestro cómo se crean varios tipos de constraints al momento de crear una tabla y también cómo añadirlos a una tabla existente.

Tipos de Constraints

1. **PRIMARY KEY**
2. **FOREIGN KEY**
3. **UNIQUE**
4. **NOT NULL**
5. **CHECK**
6. **DEFAULT**

Explicación de los Constraints

PRIMARY KEY: Garantiza que cada valor en la columna (o conjunto de columnas) es único y no nulo.

id serial PRIMARY KEY

FOREIGN KEY: Establece una relación con otra tabla. Asegura que el valor en la columna coincide con un valor en la columna de referencia.

contratacion_fk_departamento FOREIGN KEY (departamento_id) REFERENCES departamentos(id)

UNIQUE: Garantiza que todos los valores en una columna o conjunto de columnas son únicos.

unique (nombre, departamento_id)

NOT NULL: Asegura que una columna no contenga valores nulos.

nombre varying(100) NOT NULL

CHECK: Asegura que los valores en una columna cumplen una condición especificada.

edad integer NOT NULL CHECK (edad > 0)

DEFAULT: Establece un valor predeterminado para la columna si no se especifica ningún valor.

salario numeric(10, 2) DEFAULT 0.00

Añadir Constraints a una Tabla Existente

Añadir PRIMARY KEY

Añadir Tablas empleados ADD PRIMARY KEY (id);

Añadir FOREIGN KEY

*Añadir Tablas empleados ADD CONSTRAINT fk_departamento
FOREIGN KEY (departamento_id) REFERENCES departamentos(id);*

Añadir UNIQUE

Añadir Tablas empleados ADD CONSTRAINT unique_nombre UNIQUE (nombre, departamento_id);

Añadir NOT NULL

Añadir Tablas empleados ADD CONSTRAINT nombre NOT NULL;

Añadir CHECK

Añadir Tablas empleados ADD CONSTRAINT check_edad CHECK (edad > 0); **Añadir**

DEFAULT

Añadir Tablas empleados ADD CONSTRAINT salario DEFAULT 0.00;

Ejercicio Tablas y Constraints

Usando PSQL cree las tablas country, region y city

Country

```
CREATE TABLE country (  
    id serial,  
    name varchar(50)  
);
```

Agregue la llave principal (Primary Key)

```
ALTER TABLE country ADD PRIMARY KEY (id);
```

Ejecute los siguientes comandos:

\dt (Lista tablas)

Sequencia	Nombre	Tipo	Donato
public country	tabla	postgres	
public empleados	tabla	postgres	

\d Tabla : Permite visualizar estructura de la tabla

colloge=#\d country;	Tabla /zpublic.country			
Columna	Tipo	Ordenamiento	Nullable	Por omisi3n
-----+-----+-----+-----+-----				
id	integer		not null	
nextval('country_id_seq':regclass)				
name	character varying(50)			
colloge=#\di				
Sistema de relaciones				
Sequencia	Nombre	Tipo	Donato	Tabla
-----+-----+-----+-----+-----				
public	country_pkey	Índice	postgres	country
public	empleados_pkey	Índice	postgres	empleados

region

```
CREATE TABLE region (  
    id serial,  
    name varchar(50),  
    idcountry integer  
);  
  
ALTER TABLE region ADD PRIMARY KEY (id);  
  
ALTER TABLE region ADD CONSTRAINT fk_region_country FOREIGN KEY (idcountry) REFERENCES country(id);  
  
colloge=#\d region  
  
Tabla /zpublic.region  
  
Columna | Tipo | Ordenamiento | Nullable | Por omisi3n  
  
id | integer | | not null |  
nextval('region_id_seq':regclass)  
name | character varying(50) | | |  
idcountry | integer | | |  
-----  
ndices:  
"region_pkey" PRIMARY KEY, btree (id)
```

Restricciones de llave foránea:

```
"fk_region_country" FOREIGN KEY (id) REFERENCES country(id)
```

city

```
CREATE TABLE city (  
    id serial,  
    name varchar(50),  
    idregion integer  
);
```

Añadir tablas city AND PRIMARY KEY (id);

Añadir tablas city AND FOREIGN KEY (idregion) REFERENCES region(id);

Tabla /public.city

Columna	Tipo	Ordenamiento	Nullable	Por omisión
id	integer		not null	

nextval('city_id_seq'::regclass)

name	character varying(50)			
idregion	integer			

índices:

"city_pkey" PRIMARY KEY, btree (id)

Restricciones de llave foránea:

```
"fk_city_region" FOREIGN KEY (idregion) REFERENCES region(id)
```

Volcado de datos

Elimine las tablas country, region, city. Tenga en cuenta que la eliminacion la debe realizar respetando la relacion y las llaves foraneas.

```
DROP TABLE city;  
DROP TABLE region;  
DROP TABLE country;
```

Sistema de relaciones

Esquema	Nombre	Tipo	Dato
public	empleados	tabla	postgres
public	empleados_id_seq	secuencia	postgres

Cree las siguientes tablas

```
CREATE TABLE "public"."city" (  
    "id" int4 NOT NULL,  
    "name" text NOT NULL,  
    "countrycode" bpchar(3) NOT NULL,  
    "district" text NOT NULL,  
    "population" int4 NOT NULL CHECK (population >= 0),  
    PRIMARY KEY ("id")  
);
```

```

CREATE TABLE "public"."country" (
  "code" varchar(3) NOT NULL,
  "name" text NOT NULL,
  "continent" text NOT NULL CHECK ((continent = 'Asia'::text) OR (continent = 'South America'::text) OR (continent = 'North America'::text) OR (continent = 'Oceania'::text) OR (continent = 'Antarctica'::text) OR (continent = 'Africa'::text) OR (continent = 'Europe'::text) OR (continent = 'Central America'::text)),
  "region" text NOT NULL,
  "surfacearea" float NOT NULL CHECK (surfacearea >= (0)::double precision), "independ" int2,
  "population" int4 NOT NULL,
  "lifeexpectancy" float4,
  "gdp" numeric(10,2),
  "gdp$pd" numeric(10,2),
  "localname" text NOT NULL,
  "governmentform" text NOT NULL,
  "headofstate" text,
  "capital" int4,
  "code2" varchar(2) NOT NULL,
  PRIMARY KEY ("code")
);

CREATE TABLE "public"."countrylanguage" (
  "countrycode" varchar(3) NOT NULL,
  "language" text NOT NULL,
  "isofficial" bool NOT NULL,
  "percentage" float NOT NULL CHECK ((percentage >= (0)::double precision) AND (percentage <= (100)::double precision)),
  PRIMARY KEY ("countrycode","language")
);

```

Data Inicial : <https://gist.github.com/454db994cfff87a9ec542a1e92f21ff6.git>

Cree una tabla llamada continent

```

CREATE TABLE public.continent (
  code serial4 NOT NULL,
  name text NOT NULL,
  CONTINENT$ continent__pk PRIMARY KEY (code)
);

```

Caso de uso : Se requiere llenar la tabla continent con los continentes que se encuentran en la tabla country


```
select *
from country c;
```

	code	name	continent	region	surfacearea	indepyear	population	lifeexpectancy
1	ABW	Aruba	North America	Caribbean	193	[NULL]	103.000	78
2	AFG	Afghanistan	Asia	southern Asia	652.860	1.919	40.000.000	6
3	AGO	Angola	Africa	Central Africa	1.246.700	1.975	12.878.000	38
4	AIA	Anguilla	North America	Caribbean	96	[NULL]	8.000	76
5	ALB	Albania	Europe	southern Europe	28.748	1.912	3.401.200	71
6	AND	Andorra	Europe	southern Europe	468	1.278	78.000	83
7	ANT	Netherlands A	North America	Caribbean	800	[NULL]	217.000	74
8	ARE	United Arab E	Asia	Middle East	83.600	1.971	2.441.000	74
9	ARG	Argentina	South America	South America	2.780.400	1.816	37.032.000	75
10	ARM	Armenia	Asia	Middle East	29.800	1.991	3.520.000	66
11	ASM	American Sam	Oceania	Polynesia	199	[NULL]	68.000	75
12	ATA	Antarctica	Antarctica	Antarctica	13.120.000	[NULL]	0	[NULL]
13	ATF	French Southe	Antarctica	Antarctica	7.780	[NULL]	0	[NULL]
14	ATG	Antigua and B	North America	Caribbean	442	1.981	68.000	70
15	AUS	Australia	Oceania	Australia and Nev	7.741.220	1.901	18.886.000	79
16	AUT	Austria	Europe	Western Europe	83.859	1.918	8.091.800	77

Caso de uso : Liste los continentes de la tabla country. Los continentes no se deben repetir.

```
select distinct continent
from country c ;
```

	continent
1	Asia
2	South America
3	North America
4	Oceania
5	Antarctica
6	Africa
7	Central America
8	Europe

Ejecute el siguiente comando sql para listar e insertar los registros de continentes.

INSERT INTO continent (name)

SELECT DISTINCT continent

FROM country

ORDER BY continent ASC;

```
select code,name from continent c ;
```

	code	name
1	1	Africa
2	2	Antarctica
3	3	Asia
4	4	Central America
5	5	Europe
6	6	North America
7	7	Oceania
8	8	South America

Fecha, DateTime e Intervalos

La gestión de datos temporales es una parte fundamental en el desarrollo de aplicaciones modernas, ya que permite el registro y seguimiento preciso de eventos y transacciones a lo largo del tiempo. PostgreSQL, como uno de los sistemas de gestión de bases de datos relacionales más avanzados, proporciona un conjunto robusto de herramientas y tipos de datos para trabajar con fechas, horas y intervalos de tiempo.

En este capítulo, exploraremos a fondo los tipos de datos `date`, `timestamp` y `interval` en PostgreSQL, así como las funciones y operadores que nos permiten manipular y consultar estos datos de manera eficiente. Aprenderemos a utilizar estos tipos de datos para realizar cálculos temporales, manejar zonas horarias y aplicar intervalos de tiempo en nuestras consultas SQL.

Para este capítulo se usaran datos y estructura de una base de datos que podrá encontrar en el siguiente gist : <https://gist.github.com/d113a1025ce0b5d0285972196cf6773d.git> (La estructura y data fue extraída de la plataforma devtalles con fines educativos)

Cree una base de datos llamada `logistica`

CURRENT_DATE

Devuelve la fecha actual del sistema.

```
SELECT CURRENT_DATE;
```

CURRENT_TIME

Devuelve la hora actual del sistema.

```
SELECT CURRENT_TIME;
```

CURRENT_TIMESTAMP

Devuelve la fecha y hora actuales del sistema.

```
SELECT CURRENT_TIMESTAMP;
```

NOW()

Función equivalente a `CURRENT_TIMESTAMP`.

```
SELECT NOW();
```

AGE()

Calcula la diferencia entre dos fechas, o entre una fecha y la fecha actual si se proporciona solo una fecha.

```
SELECT AGE(DATE '2024-07-01');
```

```
SELECT AGE('2024-07-01', '2024-06-01');
```

EXTRACT()

Extrae subcampos (como año, mes, día, hora) de valores de fecha/hora.

```
extract(year,month,day,hour from timestamp '2024-07-13 14:23:45');  
-- Results: 2024
```

```
extract(month,year,month,day,hour from timestamp '2024-07-13 14:23:45');  
-- Results: 7
```

```
extract(year,month,day,hour from timestamp '2024-07-13 14:23:45');  
-- Results: 13
```