

MAKEFILE: Pasos para subir la Base de datos de forma correcta

Paso 1: entrar a nuestro GitHub y descargar el archivo correcto

brewLux Add files via upload		
ArepasDDL.sql	Add files via upload	3 days ago
ArepasDeArroz-DLL	Add files via upload	3 days ago
ChocolateDoom#2.pdf	Add files via upload	3 weeks ago
ChocolateDoom.zip	Add files via upload	3 weeks ago
PENS_respuestas_actualizado (1).pdf	Add files via upload	3 weeks ago
README.md	Update README with project description	3 weeks ago
all_merged.tsv	Add files via upload	3 weeks ago
telemetry.tsv	Add files via upload	3 weeks ago
telemetry10.tsv	Add files via upload	3 weeks ago
telemetry11.tsv	Add files via upload	3 weeks ago
View all files		

En nuestro caso, el archivo correcto es “ArepasDeArroz-DLL” que es el backup de nuestra base de datos

Paso 2: Entrar a tu PostgreSQL, en nuestro caso, el nuestro está en PgAdmin4 y verificar que no tengas una base de datos llamada “Chocolate Doom” ya que ese es el nombre de nuestra base de datos

The screenshot shows the PgAdmin4 interface with a tree view of database objects. At the top, there's a dropdown menu labeled "Filtered Rows...". The tree structure includes "Servers (2)", "PostgreSQL 17", "Databases (5)" which contains "Employees", "SuperHero-LAB03", "euro-Preparcial" (highlighted in blue), "post", and "postgres"; "Login/Group Roles", "Tablespaces (2)" containing "pg_default" and "pg_global", and "little_ubuntu".

Paso 3: entrar el bin de PostgreSQL a través de la terminal de esta manera

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\sanmi> cd "C:\Program Files\PostgreSQL\17\bin"
PS C:\Program Files\PostgreSQL\17\bin>

```

En nuestro caso utilizamos el siguiente comando:

```
cd "C:\Program Files\PostgreSQL\17\bin"
```

Paso 4: localizamos la ubicación de nuestro archivo “ArepasDeArroz-DLL”, en nuestro caso la ubicación estaba en:

```
"C:\Users\sanmi\Downloads\ArepasDeArroz-DDL (1)"
```

Paso 5: creamos la base de datos utilizando nuestra copia de seguridad, utilizando el siguiente comando

```
pg_restore -C -U postgres -d postgres "C:\Users\sanmi\Downloads\ArepasDeArroz-DDL (1)"
```

```
C:\Program Files\PostgreSQL\17\bin>pg_restore -C -U postgres -d postgres "C:\Users\sanmi\Downloads\ArepasDeArroz-DDL (1)"

C:\Program Files\PostgreSQL\17\bin>
```

como cosa importante a mencionar, se tendría que remplazar “**-U postgres**” por nuestro nombre de usuario en PostgreSQL

Paso 6: Verificar que se cargo de manera correcta la base de datos, visualizándola atreves de PgAdmin4

The screenshot shows the pgAdmin 4 interface. The left pane is the Object Explorer, displaying various database objects like Foreign Data Wrappers, Languages, Publications, Schemas, and the current schema 'doom'. The right pane shows a query window with the following SQL code:

```

1 SELECT *
2 FROM Doom.evento_telemetria
3

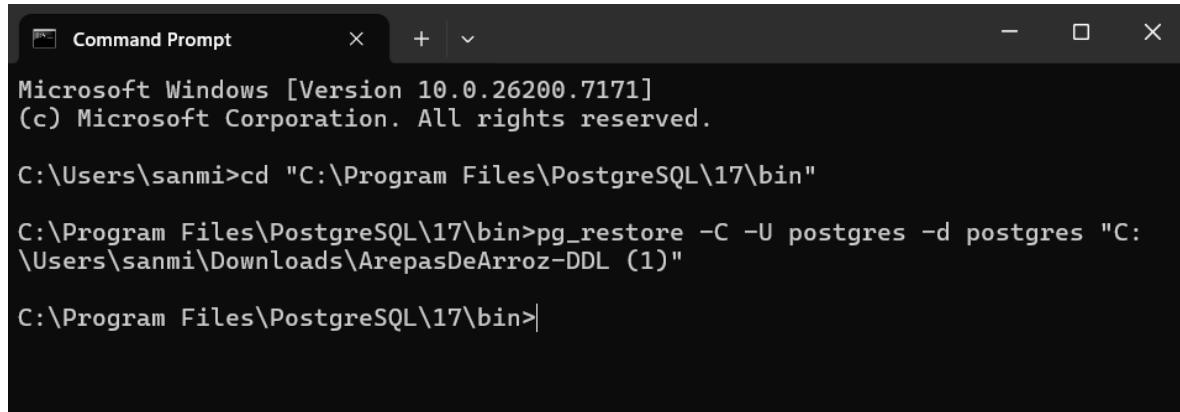
```

Below the query window is a data grid titled 'Data Output' showing the results of the query. The columns are:

	id_evento [PK] integer	id_partida integer	id_jugador integer	tic integer	posicion_x double precision	posicion_y double precision	posicion_z double precision	angulo double precision	momentum_x double precision	momentum_y double precision	momentum_z double precision
1	14	1	1	35965	2091	3682	0	261.08	-6		
2	15	1	1	60043	1168	650	-24	128.51	6		
3	16	1	1	11110	2650	3265	0	65.39	2		
4	17	1	1	2506	160	1695	56	334.42	4		
5	18	1	1	30405	1145	1810	-24	262.31	3		
6	19	1	1	3534	479	774	40	201.53	6		
7	20	1	1	52876	-65	1556	56	18.35	3		
8	21	1	1	17409	1539	1601	-24	204.04	-1		
9	22	1	1	22803	3287	2409	-144	25.66	0		
10	23	1	1	7115	2254	3354	-64	249.61	-3		
11	24	1	1	12415	740	935	-24	252.42	0		
12	25	1	1	11143	3015	2638	-144	191.6	-8		
13	26	1	1	3505	1656	1004	-40	89.65	0		
14	27	1	1	59896	-495	1546	0	95.05	-8		
15	28	1	1	20735	1070	1864	-20	259.45	1		
...
23208	23208	23208	23208	23208	23208	23208	23208	23208	23208	23208	23208

At the bottom of the pgAdmin window, it says 'Total rows: 23208 Query complete 00:00:00.150' and 'CRLF Ln 2, Col 28'. The system tray at the bottom shows icons for battery, signal, and date/time.

Demostración de los comandos:



```
Microsoft Windows [Version 10.0.26200.7171]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sanmi>cd "C:\Program Files\PostgreSQL\17\bin"

C:\Program Files\PostgreSQL\17\bin>pg_restore -C -U postgres -d postgres "C:
\Users\sanmi\Downloads\ArepasDeArroz-DDL (1)"

C:\Program Files\PostgreSQL\17\bin>
```

Ahí quedaría nuestra copia de seguridad en una base de datos.

SCRIPT DE CARGA: para ejecutar este script para la carga rápida de la base de datos, se necesita la terminal de Git (Git bash), primero se guarda el script como “setup_doom.sh” en la carpeta donde tengas ArepasDDL.sql que está en el GitHub y por ultimo desde la terminal git se hace el siguiente comando “chmod +x setup_doom.sh ./setup_doom.sh”

```
./setup_doom.sh
```

```
#!/usr/bin/env bash
```

```
# Script para recrear el esquema y cargar datos de ejemplo
```

```
# Proyecto: DOOM ArepasDeArrozPelado
```

```
# Uso:
```

```
# ./setup_doom.sh createdb -> crea la BD vacía
```

```
# ./setup_doom.sh dropdb -> elimina la BD
```

```
# ./setup_doom.sh schema -> carga el esquema (DDL)
```

```
# ./setup_doom.sh samples -> carga datos de ejemplo
```

```
# ./setup_doom.sh reset -> DROP + CREATE + schema + samples
```

```
#####
#####
```

```
DB_NAME="doomdb"
```

```
DB_USER="postgres"
```

```

DB_HOST="localhost"
DB_PORT="5432"

SCHEMA_FILE="ArepasDDL.sql" # tu DDL completo
SAMPLES_FILE="samples.sql" # solo INSERTs (si lo tienes separado)

PSQL_OPTS=""
PSQL="psql $PSQL_OPTS -h $DB_HOST -p $DB_PORT -U $DB_USER -d $DB_NAME"

usage() {
    echo "Uso: $0 {createdb|dropdb|schema|samples|reset}"
    exit 1
}

createdb_fn() {
    echo ">> Creando base de datos '$DB_NAME'..."
    createdb -h "$DB_HOST" -p "$DB_PORT" -U "$DB_USER" "$DB_NAME" \
        && echo ">> Base de datos creada." \
        || echo "[!] La BD ya existe o hubo un error."
}

dropdb_fn() {
    echo ">> Eliminando base de datos '$DB_NAME'.."
    dropdb -h "$DB_HOST" -p "$DB_PORT" -U "$DB_USER" "$DB_NAME" \
        && echo ">> Base de datos eliminada." \
        || echo "[!] La BD no existe o hubo un error."
}

schema_fn() {
    if [ ! -f "$SCHEMA_FILE" ]; then
        echo "[ERROR] No se encontró el archivo '$SCHEMA_FILE'"
        exit 1
}

```

```

fi

echo ">> Cargando esquema desde '$SCHEMA_FILE'..."
$PSQL -f "$SCHEMA_FILE"
echo ">> Esquema cargado correctamente."
}

samples_fn() {
if [ ! -f "$SAMPLES_FILE" ]; then
echo "[WARN] No se encontró '$SAMPLES_FILE'"
echo "    Si tu DDL ya incluye los INSERTs, puedes omitir este paso."
return 0
fi

echo ">> Cargando datos de ejemplo desde '$SAMPLES_FILE'..."
$PSQL -f "$SAMPLES_FILE"
echo ">> Datos de ejemplo cargados."
}

reset_fn() {
dropdb_fn
createdb_fn
schema_fn
samples_fn

echo ">> Reset completo de la base de datos '$DB_NAME'."

}

[ $# -lt 1 ] && usage

case "$1" in
createdb) createdb_fn ;;
dropdb) dropdb_fn ;;
schema) schema_fn ;;

```

```
samples) samples_fn ;;

reset) reset_fn ;;

*) usage ;;

esac
```

después de ejecutar el script se pueden utilizar los siguientes comandos para manipular la base de datos desde la Git Bash:

./setup_doom.sh createdb: Crea la base de datos y las estructuras iniciales

./setup_doom.sh schema: Crea las tablas, relaciones, constraints...

./setup_doom.sh samples: Inserta datos de ejemplo

./setup_doom.sh reset: Borra todo y vuelve a crear la base desde cero