IIC2413

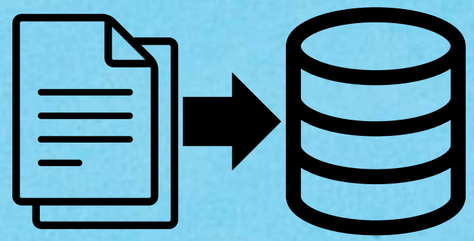
AYUDANTÍA 5

OBJETIVOS

- Conectarse por SSH al servidor
- Manejar archivos en servidor Ubuntu
- Cargar archivos *.csv a un esquema.
- Crear y ejecutar archivos *.sql
- Realizar consultas a la Base de Datos



ubuntu





BIENVENIDOS A

PSQL

¿Qué es? psql es una interfaz de terminal para PostgreSQL .

¿Para qué sirve? Permite escribir consultas de forma interactiva, enviarlas a PostgreSQL y ver los resultados. También se puede ingresar información desde un archivo o mediante argumentos de línea de comandos.

POSTGRESQL

- **¿Qué es?** Sistema de gestión de bases de datos relacionales (RDBMS) de código abierto.
- **¿Para qué sirve?** Almacena, organiza y gestiona datos complejos de forma segura.
- **Casos de uso:** Ideal para aplicaciones web, análisis de datos y sistemas empresariales.
- **Nuestro entorno de trabajo:** Se encuentra instalado y ejecutándose en el servidor de Ubuntu (stonebraker).



SERVIDOR

ubuntu

CONECTARSE A SERVIDOR

En CMD/Terminal ejecutar:

```
ssh usuarioUC.e1@stonebraker.ing.uc.cl
```

usuarioUC -> usuarioUC~~@uc.cl~~

e# → # = número_de_entrega

Ej: Si mi correo es asierra2@uc.cl

```
ssh asierra2.e1@stonebraker.ing.uc.cl
```



SUBIR ARCHIVO A SERVIDOR

En CMD ejecutar:

```
scp {nombre_archivo} {usuario_uc}.e1@stonebraker.ing.uc.cl:{ruta_destino}
```

Debes estar en la ubicación del archivo {nombre_archivo}, si no debes incorporar la ruta {ruta_archivo\nombre_archivo}

Ejemplo subida archivo a servidor:

```
C:\Users\antos\OneDrive\Desktop>scp bdd.txt asierra2.e1@stonebraker.ing.uc.cl:E1
asierra2.e1@stonebraker.ing.uc.cl's password:
bdd.txt                                       100%    5     0.5KB/s   00:00
```

```
C:\Users\antos\OneDrive>scp Desktop\bdd.txt asierra2.e1@stonebraker.ing.uc.cl:E1
asierra2.e1@stonebraker.ing.uc.cl's password:
bdd.txt                                       100%    5     0.5KB/s   00:00
```

COMANDOS LINUX/UBUNTU

- ls (listar)
- mkdir (crear carpeta)
- touch (crear archivo)
- nano (editor integrado en la terminal)
- rm (eliminar archivo)
- rm –r (eliminar carpeta)
- cd (cambiar de directorio)
- pwd (muestra la ruta del directorio de trabajo actual)
- cat (muestra contenido del archivo)

ubuntu

CONECTARSE A SERVIDOR

```
C:\Users\Max Santibáñez>ssh max.santibanez.e1@stonebraker.ing.uc.cl
max.santibanez.e1@stonebraker.ing.uc.cl's password:
```

```
Welcome to Ubuntu 24.04.4 LTS (GNU/Linux 6.8.0-107-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Thu Apr 16 10:02:20 PM -04 2026

  System load:  0.16                Processes:               393
  Usage of /:   12.9% of 180.99GB   Users logged in:         1
  Memory usage: 4%                  IPv4 address for ens34: 146.155.13.76
  Swap usage:   0%
```

```
max.santibanez.e1@stonebreaker:~$
```

LISTAR ARCHIVOS

```
max.santibanez.e1@stonebreaker:~$ ls
E1
max.santibanez.e1@stonebreaker:~$ cd E1
max.santibanez.e1@stonebreaker:~/E1$
```

```
max.santibanez.e1@stonebreaker:~/E1$ ls
bdd.txt
```

CREAR Y ELIMINAR ARCHIVOS/ CARPETAS

- Crear:

```
asierra2.e1@stonebreaker:~/E1$ mkdir Ayudantia
asierra2.e1@stonebreaker:~/E1$ touch ejercicio.sql
asierra2.e1@stonebreaker:~/E1$ ls
Ayudantia  ayudantia.sql  bdd.txt  ejercicio.sql
```

- Eliminar:

```
asierra2.e1@stonebreaker:~/E1$ rm bdd.txt
asierra2.e1@stonebreaker:~/E1$ rm -r Ayudantia/
asierra2.e1@stonebreaker:~/E1$ ls
ayudantia.sql  ejercicio.sql
```

COMANDOS

PSQL

COMANDOS IMPORTANTES

1. Por defecto para conectarse a postgres se utiliza el siguiente comando

```
psql
```

Si no funciona tratar con

```
psql postgres
```

4. Para ejecutar un archivo SQL

```
psql -f archivo.sql
```

COMANDOS IMPORTANTES

1. Para crear base de datos

```
CREATE DATABASE nombre_db;
```

2. Para conectarse a una base de datos dentro de postgres

```
\c nombre_db
```

3. Para ver todas las bases de datos de tu equipo

```
\l
```

4. Para ver las tablas de la base de datos actual

```
\dt
```

COMANDOS IMPORTANTES

5. Para ver los atributos que recibe y las IC de las tablas

```
\d nombre_tabla
```

6. Para salir de postgres

```
\q
```

7. Para crear usuarios

```
CREATE USER nombre_usuario WITH PASSWORD 'contraseña';
```

9. Para visualizar los usuarios de tu equipo

```
\du
```

EJEMPLO

Esquema relacional:

- **jugadores**(sofifa_id, nombre, nombre_completo, precio_eur, edad, nacimiento, altura, peso, club_id, liga_id, numero, fecha_ingreso, nacionalidad_id, pie)
- **ligas**(liga_id, nombre)
- **clubes**(club_id, nombre, liga_id)
- **nacionalidades**(nacionalidad_id, nombre)
- **posiciones**(posicion_id, nombre)
- **jugador_posiciones**(jugador_id, posicion_id)

CREAR TABLAS

- Crear archivo esquema.sql en local o en servidor

```
max.santibanez.e1@stonebreaker:~/E1$ nano esquema.sql
```

- Por si ya fue creado y ejecutado anteriormente colocar al principio (cuidado, borra las tablas)

```
-- Borrar tablas si existen para poder reejecutar el script sin errores
DROP TABLE IF EXISTS jugador_posiciones CASCADE;
DROP TABLE IF EXISTS posiciones CASCADE;
DROP TABLE IF EXISTS jugadores CASCADE;
DROP TABLE IF EXISTS clubes CASCADE;
DROP TABLE IF EXISTS ligas CASCADE;
DROP TABLE IF EXISTS nacionalidades CASCADE;
```

- Dentro del archivo esquema.sql comenzar a definir nuestras tablas

CREAR TABLAS

- esquema.sql debe contener:

```sql
CREATE TABLE ligas (
    liga_id SERIAL PRIMARY KEY,
    nombre VARCHAR(100) UNIQUE NOT NULL
);

CREATE TABLE nacionalidades (
    nacionalidad_id SERIAL PRIMARY KEY,
    nombre VARCHAR(100) UNIQUE NOT NULL
);

CREATE TABLE posiciones (
    posicion_id SERIAL PRIMARY KEY,
    nombre VARCHAR(10) UNIQUE NOT NULL
);
```

```sql
CREATE TABLE clubes (
    club_id SERIAL PRIMARY KEY,
    nombre VARCHAR(100) UNIQUE NOT NULL,
    liga_id INT REFERENCES ligas(liga_id)
);

CREATE TABLE jugadores (
    sofifa_id INTEGER PRIMARY KEY,
    nombre TEXT,
    nombre_completo TEXT,
    precio_eur NUMERIC,
    edad INTEGER,
    nacimiento DATE,
    altura INTEGER,
    peso INTEGER,
    club_id INT REFERENCES clubes(club_id),
    liga_id INT REFERENCES ligas(liga_id),
    numero INTEGER,
    fecha_ingreso DATE,
    nacionalidad_id INT REFERENCES nacionalidades(nacionalidad_id),
    pie TEXT
);

CREATE TABLE jugador_posiciones (
    jugador_id INT,
    posicion_id INT,
    PRIMARY KEY (jugador_id, posicion_id),
    FOREIGN KEY (jugador_id) REFERENCES jugadores(sofifa_id),
    FOREIGN KEY (posicion_id) REFERENCES posiciones(posicion_id)
);
```

EJECUTAR Y VER ESQUEMA

- Ejecutar script .sql con psql -f

```
max.santibanez.e1@stonebreaker:~/E1$ ls
esquema.sql
max.santibanez.e1@stonebreaker:~/E1$ psql -f esquema.sql
psql:esquema.sql:2: NOTICE:  table "jugador_posiciones" does not exist, skipping
DROP TABLE
psql:esquema.sql:3: NOTICE:  table "posiciones" does not exist, skipping
DROP TABLE
psql:esquema.sql:4: NOTICE:  table "jugadores" does not exist, skipping
DROP TABLE
psql:esquema.sql:5: NOTICE:  table "clubes" does not exist, skipping
DROP TABLE
psql:esquema.sql:6: NOTICE:  table "ligas" does not exist, skipping
DROP TABLE
psql:esquema.sql:7: NOTICE:  table "nacionalidades" does not exist, skipping
DROP TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
```

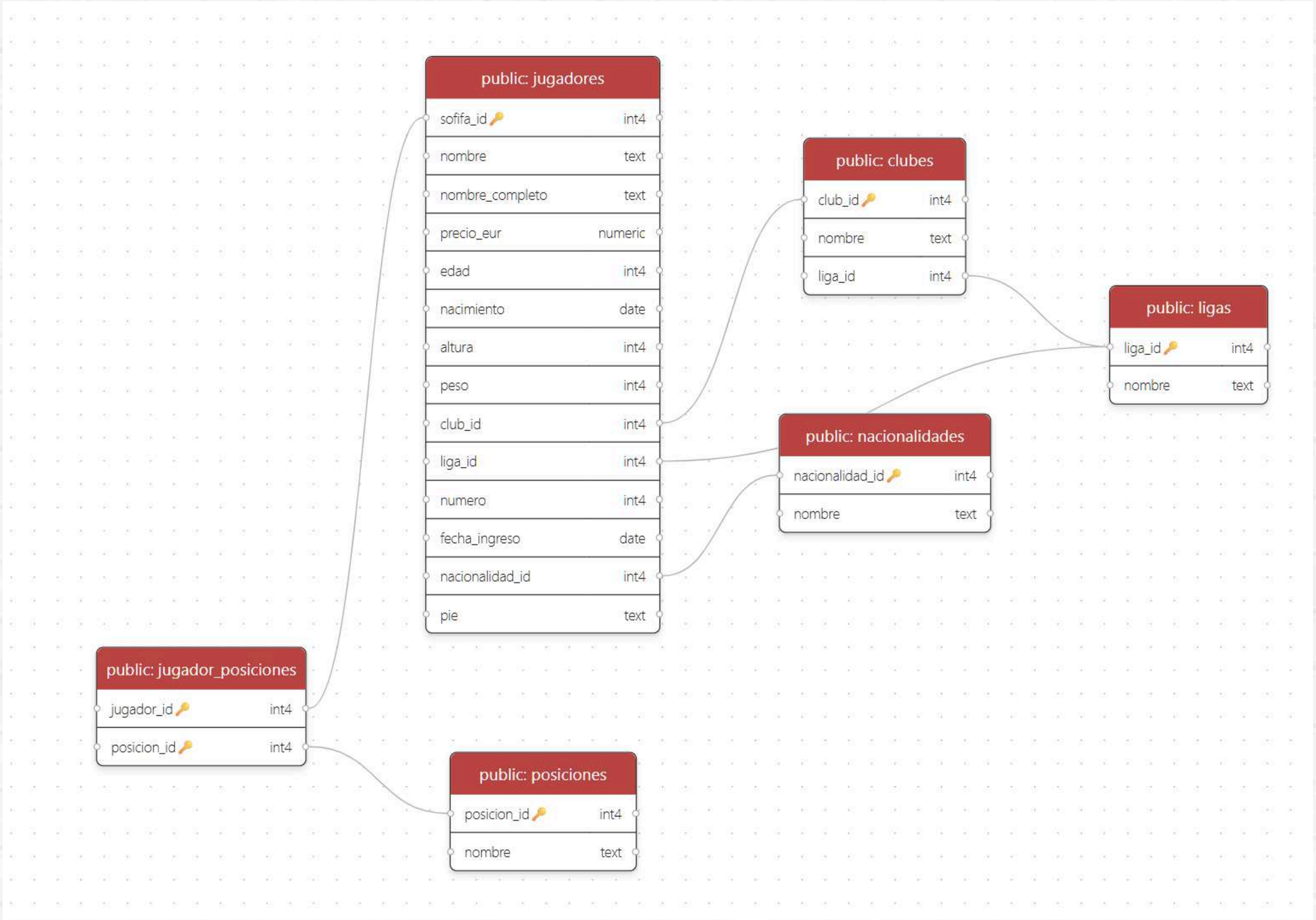
EJECUTAR Y VER ESQUEMA

- Listar tablas con \d

```
max.santibanez.e1@stonebreaker:~/E1$ psql max.santibanez.e1
psql (16.13 (Ubuntu 16.13-0ubuntu0.24.04.1))
Type "help" for help.

max.santibanez.e1=>
```

```
max.santibanez.e1=> \d
                    List of relations
 Schema |         Name         | Type  |        Owner
--------+----------------------+-------+---------------------
 public | clubes               | table | max.santibanez.e1
 public | jugador_posiciones   | table | max.santibanez.e1
 public | jugadores            | table | max.santibanez.e1
 public | ligas                | table | max.santibanez.e1
 public | nacionalidades       | table | max.santibanez.e1
 public | posiciones           | table | max.santibanez.e1
```


public: jugadores
sofifa_id int4
nombre text
nombre_completo text
precio_eur numeric
edad int4
nacimiento date
altura int4
peso int4
club_id int4
liga_id int4
numero int4
fecha_ingreso date
nacionalidad_id int4
pie text
public: clubes
club_id int4
nombre text
liga_id int4
public: ligas
liga_id int4
nombre text
public: nacionalidades
nacionalidad_id int4
nombre text
public: jugador_posiciones
jugador_id int4
posicion_id int4
public: posiciones
posicion_id int4
nombre text

CARGAR DATOS DESDE CSV

- Crear archivo poblar_tablas.sql (nano)
- Tabla temporal para asignar dominio de cada atributo:

```sql
-- Se crea una tabla temporal para la carga inicial
CREATE TEMP TABLE jugadores_tmp (
    sofifa_id INTEGER,
    nombre TEXT,
    nombre_completo TEXT,
    posiciones TEXT,
    precio_eur NUMERIC,
    edad INTEGER,
    nacimiento DATE,
    altura INTEGER,
    peso INTEGER,
    club TEXT,
    liga TEXT,
    numero INTEGER,
    fecha_ingreso DATE,
    nacionalidad TEXT,
    pie TEXT
);
```

CARGAR DATOS DESDE CSV

- El argumento CSV HEADER ignora la primera línea del archivo, y los datos se cargan en el orden que indicamos en el paréntesis
- Carga de datos archivo .csv (debe estar en la carpeta actual) con \copy:

```
-- Se cargan los datos desde el CSV
-- (asegurarse que el archivo esté en la misma carpeta)
\copy jugadores_tmp(sofifa_id,
                    nombre,
                    nombre_completo,
                    posiciones,
                    precio_eur,
                    edad,
                    nacimiento,
                    altura,
                    peso,
                    club,
                    liga,
                    numero,
                    fecha_ingreso,
                    nacionalidad,
                    pie)
                    FROM 'players_22.csv' DELIMITER ';' CSV HEADER;
```

POBLAR TABLAS

- Se crea la tabla datos con llave primaria:

```sql
-- Se elimina la tabla datos si es que fue creada antes para reejecución del script
DROP TABLE IF EXISTS datos CASCADE;

-- Se crea la tabla consolidada 'datos' para validar la información
CREATE TABLE datos (
    sofifa_id INTEGER PRIMARY KEY,
    nombre TEXT,
    nombre_completo TEXT,
    posiciones TEXT,
    precio_eur NUMERIC,
    edad INTEGER,
    nacimiento DATE,
    altura INTEGER,
    peso INTEGER,
    club TEXT,
    liga TEXT,
    numero INTEGER,
    fecha_ingreso DATE,
    nacionalidad TEXT,
    pie TEXT
);
```

- Insertamos los atributos de la tabla temporal a tabla datos

```sql
-- Se extraen solo los datos válidos para insertar en la tabla datos
INSERT INTO datos (sofifa_id,
                   nombre,
                   nombre_completo,
                   posiciones,
                   precio_eur,
                   edad,
                   nacimiento,
                   altura,
                   peso,
                   club,
                   liga,
                   numero,
                   fecha_ingreso,
                   nacionalidad,
                   pie )
    SELECT
    sofifa_id,
    nombre,
    nombre_completo,
    posiciones,
    precio_eur,
    edad,
    nacimiento,
    altura,
    peso,
    club,
    liga,
    numero,
    fecha_ingreso,
    nacionalidad,
    pie
    FROM jugadores_tmp
    WHERE sofifa_id IS NOT NULL
    AND nombre_completo IS NOT NULL;
```

- Se puebla cada tabla (ligas, clubes y nacionalidades) con valores únicos

```sql
-- Poblar tablas con datos únicos para ligas, clubes y nacionalidades
INSERT INTO ligas (nombre)
SELECT DISTINCT liga
FROM datos
WHERE liga IS NOT NULL;

INSERT INTO clubes (nombre, liga_id)
SELECT DISTINCT d.club, l.liga_id
FROM datos d
JOIN ligas l ON d.liga = l.nombre
WHERE d.club IS NOT NULL;

INSERT INTO nacionalidades (nombre)
SELECT DISTINCT nacionalidad
FROM datos
WHERE nacionalidad IS NOT NULL;
```

- Se puebla la tabla principal jugadores y se hace LEFT JOIN con otras tablas para mantener llaves foráneas (IDs)

```sql
-- Poblar tabla principal de jugadores recuperando las llaves foráneas (IDs) mediante LEFT JOINS
INSERT INTO jugadores (sofifa_id, nombre, nombre_completo, precio_eur, edad, nacimiento, altura, peso, club_id, liga_id, numero, fecha_ingreso, nacionalidad_id, pie
SELECT d.sofifa_id,
       d.nombre,
       d.nombre_completo,
       d.precio_eur,
       d.edad,
       d.nacimiento,
       d.altura,
       d.peso,
       c.club_id,
       l.liga_id,
       d.numero,
       d.fecha_ingreso,
       n.nacionalidad_id,
       d.pie
FROM datos d
LEFT JOIN clubes c ON d.club = c.nombre
LEFT JOIN ligas l ON d.liga = l.nombre
LEFT JOIN nacionalidades n ON d.nacionalidad = n.nombre;
```

- Se separan las posiciones de cada jugador (que vienen juntas como texto) en filas independientes
- Para cada jugador, se asignan sus posiciones

```sql
-- Tablas de posiciones (Manejo de strings y arreglos)
INSERT INTO posiciones (nombre)
SELECT DISTINCT unnest(string_to_array(posiciones, ', ')) AS pos
FROM datos
WHERE posiciones IS NOT NULL;

-- Relación muchos a muchos entre jugadores y posiciones
INSERT INTO jugador_posiciones (jugador_id, posicion_id)
SELECT j.sofifa_id, p.posicion_id
FROM datos d
JOIN jugadores j ON d.sofifa_id = j.sofifa_id
JOIN posiciones p ON p.nombre = ANY(string_to_array(d.posiciones, ', '))
WHERE d.posiciones IS NOT NULL;
```

player_positions
RW, ST, CF
ST
ST, LW
LW, CAM
CM, CAM
GK

CONSULTAS

PostgreSQL

CONSULTAS

1. Top 5 jugadores con mayor número de posiciones distintas. En caso de empate se muestran todos los jugadores con igual cantidad.

Creamos el archivo consulta_1.sql dentro del cual escribimos la consulta 1.

```sql
WITH ranking AS (
    SELECT
        j.sofifa_id,
        j.nombre                          AS jugador,
        COUNT(DISTINCT jp.posicion_id)    AS cantidad_posiciones
    FROM jugadores j
    JOIN jugador_posiciones jp ON j.sofifa_id = jp.jugador_id
    GROUP BY j.sofifa_id, j.nombre
),
top_n AS (
    SELECT cantidad_posiciones
    FROM ranking
    ORDER BY cantidad_posiciones DESC
    LIMIT 5
)
SELECT
    r.jugador,
    r.cantidad_posiciones
FROM ranking r
WHERE r.cantidad_posiciones >= (SELECT MIN(cantidad_posiciones) FROM top_n)
ORDER BY r.cantidad_posiciones DESC, r.jugador;
```

CONSULTAS

1. Top 5 jugadores con mayor número de posiciones distintas. En caso de empate se muestran todos los jugadores con igual cantidad.

Podemos ejecutar el archivo consulta_1.sql de dos formas:

1. Desde la consola de psql

```
ayudantia=# \i consulta_1.sql
```

2. Desde terminal de Ubuntu (local o servidor)

```
antosierra@AntitoNotebook:~/ayudantia$ psql ayudantia -f consulta_1.sql
```

CONSULTAS

1. Top 5 jugadores con mayor número de posiciones distintas. En caso de empate se muestran todos los jugadores con igual cantidad.

Al ejecutar consulta_1.sql se muestra el resultado en consola.

jugador	cantidad_posiciones
A Lan	3
A. Abdellaoui	3
A. Abdi	3
A. Abdijanovic	3
A. Acosta	3
A. Acosta	3
A. Aganović	3
A. Ait El Hadj	3
A. Al Badaoui	3
A. Al Dhafeeri	3
A. Al Ghamdi	3
A. Al Haider	3
A. Al Meqran	3
A. Al Qahtani	3
A. Al Shammari	3
A. Al Suwat	3
A. Al Zaqan	3
A. Amade	3
A. Andrade	3
A. Androutsos	3
A. Aouchiche	3
A. Appiah	3
A. Arigoni	3
A. Arslan	3

:

CONSULTAS

2. Top 5 ligas con mayor valor total de mercado de sus jugadores

Creamos el archivo consulta_2.sql dentro del cual escribimos la consulta 2.

```sql
WITH ranking AS (
    SELECT
        l.nombre                    AS liga,
        SUM(j.precio_eur)           AS valor_total_jugadores
    FROM jugadores j
    JOIN ligas l ON j.liga_id = l.liga_id
    WHERE j.precio_eur IS NOT NULL AND j.precio_eur > 0
    GROUP BY l.nombre
),
top_n AS (
    SELECT valor_total_jugadores
    FROM ranking
    ORDER BY valor_total_jugadores DESC
    LIMIT 10
)
SELECT
    r.liga,
    r.valor_total_jugadores
FROM ranking r
WHERE r.valor_total_jugadores >= (SELECT MIN(valor_total_jugadores) FROM top_n)
ORDER BY r.valor_total_jugadores DESC;
```

CONSULTAS

2. Top 10 ligas con mayor valor total de mercado de sus jugadores

Ejecutamos el archivo consulta_2.sql:

```
ayudantia=# \i consulta_2.sql
             liga                | valor_total_jugadores
---------------------------------+-----------------------
 English Premier League          |           9148980000.0
 Spain Primera Division          |           7282200000.0
 Italian Serie A                 |           5308695000.0
 German 1. Bundesliga            |           4763380000.0
 French Ligue 1                  |           4239495000.0
 Portuguese Liga ZON SAGRES      |           2015385000.0
 Holland Eredivisie              |           1454925000.0
 Argentina Primera División      |           1430055000.0
 Campeonato Brasileiro Série A   |           1394130000.0
 English League Championship     |           1261620000.0
(10 rows)
```

CONSULTAS

3a. Top 5 posiciones más frecuentes entre los jugadores

3b. Top 5 ligas con mayor número de jugadores registrados

Creamos el archivo consulta_3.sql dentro del cual escribimos la consulta 3.

```sql
-- Top 5 posiciones más frecuentes entre los jugadores

SELECT
    p.nombre        AS posicion,
    COUNT(*)        AS cantidad_jugadores
FROM jugador_posiciones jp
JOIN posiciones p ON jp.posicion_id = p.posicion_id
GROUP BY p.nombre
ORDER BY cantidad_jugadores DESC
LIMIT 5;

-- Top 5 ligas con mayor número de jugadores registrados

SELECT
    l.nombre        AS liga,
    COUNT(*)        AS cantidad_jugadores
FROM jugadores j
JOIN ligas l ON j.liga_id = l.liga_id
GROUP BY l.nombre
ORDER BY cantidad_jugadores DESC
LIMIT 5;
```

CONSULTAS

3a. Top 5 posiciones más frecuentes entre los jugadores

3b. Top 5 ligas con mayor número de jugadores registrados

Ejecutamos el archivo consulta_3.sql:

```
ayudantia=# \i consulta_3.sql
 posicion | cantidad_jugadores
----------+--------------------
 CM       |               4204
 CB       |               4021
 ST       |               3398
 CDM      |               3014
 LM       |               2521
(5 rows)

            liga              | cantidad_jugadores
------------------------------+--------------------
 USA Major League Soccer      |                771
 Argentina Primera División   |                727
 English League Championship  |                717
 English Premier League       |                652
 Spain Primera Division       |                633
(5 rows)
```

CONSULTAS

4a. Número mensual de incorporaciones de jugadores al sistema

4b. Top 5 clubes con más incorporaciones históricas

4c. Top 5 ligas con más incorporaciones históricas

Creamos el archivo consulta_4.sql dentro del cual escribimos la consulta 4.

```sql
-- A) Incorporaciones mensuales
SELECT
    TO_CHAR(j.fecha_ingreso, 'YYYY-MM') AS mes,
    COUNT(*)                            AS total_incorporaciones
FROM jugadores j
WHERE j.fecha_ingreso IS NOT NULL
GROUP BY mes
ORDER BY mes;

-- B) Top 5 clubes con más incorporaciones históricas
SELECT
    c.nombre  AS club,
    COUNT(*)  AS total_incorporaciones
FROM jugadores j
JOIN clubes c ON j.club_id = c.club_id
WHERE j.fecha_ingreso IS NOT NULL
GROUP BY c.nombre
ORDER BY total_incorporaciones DESC
LIMIT 5;

-- C) Top 5 ligas con más incorporaciones históricas
SELECT
    l.nombre  AS liga,
    COUNT(*)  AS total_incorporaciones
FROM jugadores j
JOIN ligas l ON j.liga_id = l.liga_id
WHERE j.fecha_ingreso IS NOT NULL
GROUP BY l.nombre
ORDER BY total_incorporaciones DESC
LIMIT 5;
```

CONSULTAS

4a. Número mensual de incorporaciones de jugadores al sistema

4b. Top 5 clubes con más incorporaciones históricas

4c. Top 5 ligas con más incorporaciones históricas

Ejecutamos el archivo consulta_4.sql:

```
ayudantia=# \i consulta_4.sql
           club            | total_incorporaciones
---------------------------+-----------------------
 Sevilla FC                |                    33
 Borussia Mönchengladbach  |                    33
 Everton                   |                    33
 Burnley                   |                    33
 Brighton & Hove Albion    |                    33
(5 rows)

            liga             | total_incorporaciones
-----------------------------+-----------------------
 USA Major League Soccer     |                   747
 English League Championship |                   668
 Argentina Primera División  |                   652
 English Premier League      |                   632
 Spain Primera Division      |                   588
(5 rows)
```

```
   mes    | total_incorporaciones
----------+-----------------------
 2002-01  |                     1
 2002-08  |                     1
 2003-01  |                     1
 2003-07  |                     1
 2003-12  |                     1
 2004-01  |                     1
 2004-07  |                     2
 2004-08  |                     1
 2005-01  |                     5
 2005-06  |                     1
 2005-07  |                     3
 2005-09  |                     1
 2006-01  |                     4
 2006-07  |                     2
 2006-08  |                     1
 2007-01  |                     6
 2007-02  |                     1
 2007-05  |                     1
 2007-07  |                     3
 2007-08  |                     1
 2008-01  |                     7
 2008-06  |                     1
 2008-07  |                     9
 2008-08  |                     1
:
```

CONSULTAS

5. Top 25 jugadores que juegan en más de una posición, en orden descendiente segun precio de mercado y en caso de empate compara por edad ascendente.

Creamos el archivo consulta_5.sql dentro del cual escribimos la consulta 5.

```sql
SELECT j.nombre AS nombre_jugador,
       c.nombre AS club,
       l.nombre AS liga,
       j.edad,
       j.precio_eur,
       COUNT(p.posicion_id) AS cantidad_posiciones
FROM jugadores j
JOIN clubes c
  ON j.club_id = c.club_id
JOIN ligas l
  ON c.liga_id = l.liga_id
JOIN jugador_posiciones jp
  ON j.sofifa_id = jp.jugador_id
JOIN posiciones p
  ON jp.posicion_id = p.posicion_id
WHERE precio_eur > 0
GROUP BY j.sofifa_id, j.nombre, c.nombre, l.nombre, j.edad, j.precio_eur
HAVING COUNT(p.posicion_id) > 1
ORDER BY j.precio_eur DESC,
         j.edad ASC
LIMIT 25;
```

CONSULTAS

5. Top 25 jugadores que juegan en más de una posición, en orden descendiente segun precio de mercado y en caso de empate compara por edad ascendente.

Ejecutamos el archivo consulta_5.sql:

nombre_jugador	club	liga	edad	precio_eur	cantidad_posiciones
K. Mbappé	Paris Saint-Germain	French Ligue 1	22	194000000.0	2
Neymar Jr	Paris Saint-Germain	French Ligue 1	29	129000000.0	2
K. De Bruyne	Manchester City	English Premier League	30	125500000.0	2
F. de Jong	FC Barcelona	Spain Primera Division	24	119500000.0	3
J. Sancho	Manchester United	English Premier League	21	116500000.0	3
J. Kimmich	FC Bayern München	German 1. Bundesliga	26	108000000.0	2
R. Sterling	Manchester City	English Premier League	26	107500000.0	2
H. Son	Tottenham Hotspur	English Premier League	28	104000000.0	3
N. Kanté	Chelsea	English Premier League	30	100000000.0	2
P. Foden	Manchester City	English Premier League	21	94500000.0	3
K. Havertz	Chelsea	English Premier League	22	94500000.0	3
L. Goretzka	FC Bayern München	German 1. Bundesliga	26	93000000.0	2
P. Dybala	Juventus	Italian Serie A	27	93000000.0	2
Marquinhos	Paris Saint-Germain	French Ligue 1	27	90500000.0	2
Marcos Llorente	Atlético de Madrid	Spain Primera Division	26	88000000.0	3
João Félix	Atlético de Madrid	Spain Primera Division	21	82000000.0	2
K. Coman	FC Bayern München	German 1. Bundesliga	25	81000000.0	3
F. Chiesa	Juventus	Italian Serie A	23	80500000.0	3
Bernardo Silva	Manchester City	English Premier League	26	80000000.0	3
M. Verratti	Paris Saint-Germain	French Ligue 1	28	79500000.0	2
P. Pogba	Manchester United	English Premier League	28	79500000.0	2
L. Messi	Paris Saint-Germain	French Ligue 1	34	78000000.0	3
M. Rashford	Manchester United	English Premier League	23	77500000.0	2
Oyarzabal	Real Sociedad	Spain Primera Division	24	77500000.0	2

CONSULTAS

6. Top 10 de equipos con mas jugadores nacidos el mismo mes.

Creamos el archivo consulta_6.sql dentro del cual escribimos la consulta 6.

```sql
SELECT c.nombre AS club,
       TO_CHAR(j.nacimiento, 'MM') AS mes_nacimiento,
       COUNT(*) AS cantidad_jugadores
FROM jugadores j
JOIN clubes c
  ON j.club_id = c.club_id
WHERE j.nacimiento IS NOT NULL
GROUP BY c.nombre, mes_nacimiento
ORDER BY cantidad_jugadores DESC;
```

CONSULTAS

6. Top 10 de equipos con mas jugadores nacidos el mismo mes.

Ejecutamos el archivo consulta_6.sql:

club	mes_nacimiento	cantidad_jugadores
Palmeiras	02	20
Atlético Clube Goianiense	02	20
RB Bragantino	02	20
Clube Atlético Mineiro	02	20
Bahia	02	20
Santos	02	20
Cuiabá	02	20
Ceará Sporting Club	02	20
Grêmio	02	20
Juventude	02	20
São Paulo	02	20
Associação Chapecoense de Futebol	02	20
Internacional	02	20
Fortaleza	02	20
Fluminense	02	20
Sport Club Corinthians Paulista	02	20
Club Athletico Paranaense	02	20
Flamengo	02	20
Sunderland	04	10
Venezia FC	05	10
LOSC Lille	01	9
Borussia Dortmund	01	9
Tianjin Jinmen Tiger FC	01	9
Hull City	12	9

¡GRACIAS!