2023

Joaquín García Gutiérrez

IES ALIXAR

6-11-2023



Proyecto de BD eSports CSGO

Contenido

[**1.** **Introducción** 3](#_Toc163378408)

[*Versión 0.1* 3](#_Toc163378409)

[*Versión 0.2* 3](#_Toc163378410)

[*Version 0.3* 3](#_Toc163378411)

[*Version 0.4* 3](#_Toc163378412)

[**2.** **Descripción** 3](#_Toc163378413)

[**3.** **Modelo Entidad/Relación extendido y Modelo relacional.** 5](#_Toc163378414)

[1. MER. 5](#_Toc163378415)

[2. Modelo relacional. 5](#_Toc163378416)

[**4.** **Paso a tablas en MySQL Workbench.** 7](#_Toc163378417)

[**5.** **Importado a MySQL** 7](#_Toc163378418)

[**6.** **Actualizaciones y modificaciones** 7](#_Toc163378419)

[*Modelo relacional:* 7](#_Toc163378420)

[**7.** **Carga de datos masiva** 8](#_Toc163378421)

[**1.** **Tutorial de importación de csv para la carga masiva de datos.** 8](#_Toc163378422)

[**8.** **Consultas SQL** 13](#_Toc163378423)

[*Objetivo:* 13](#_Toc163378424)

[*SLQ:* 13](#_Toc163378425)

[*Resultado:* 13](#_Toc163378426)

[*Objetivo:* 14](#_Toc163378427)

[*SLQ 2:* 14](#_Toc163378428)

[*Resultado:* 14](#_Toc163378429)

[*Objetivo:* 15](#_Toc163378430)

[*SLQ 3:* 15](#_Toc163378431)

[*Resultado:* 15](#_Toc163378432)

[*Objetivo:* 15](#_Toc163378433)

[*SLQ 4:* 16](#_Toc163378434)

[*Resultado:* 16](#_Toc163378435)

[*Objetivo:* 16](#_Toc163378436)

[*SLQ 5:* 16](#_Toc163378437)

[*Resultado:* 17](#_Toc163378438)

[**9.** **Vistas** 17](#_Toc163378439)

[*Vista de la consulta nº 1.* 17](#_Toc163378440)

[SQL: 17](#_Toc163378441)

[Resultado: 18](#_Toc163378442)

[*Vista de la consulta nº 4.* 18](#_Toc163378443)

[SQL: 18](#_Toc163378444)

[Resultado: 19](#_Toc163378445)

[**10.** **Funciones y procedimientos** 19](#_Toc163378446)

[*Función 1:* 19](#_Toc163378447)

[Objetivo: 19](#_Toc163378448)

[SQL: 19](#_Toc163378449)

[Resultado: 20](#_Toc163378450)

[*Función 2:* 20](#_Toc163378451)

[Objetivo: 20](#_Toc163378452)

[SQL: 20](#_Toc163378453)

[Resultado: 22](#_Toc163378454)

[*Procedimiento 1:* 22](#_Toc163378455)

[Objetivo: 22](#_Toc163378456)

[SQL: 22](#_Toc163378457)

[Resultado: 23](#_Toc163378458)

[*Procedimiento 2:* 23](#_Toc163378459)

[Objetivo: 23](#_Toc163378460)

[SQL: 23](#_Toc163378461)

[Resultado: 25](#_Toc163378462)

[*Procedimiento 3:* 25](#_Toc163378463)

[Objetivo: 25](#_Toc163378464)

[SQL: 25](#_Toc163378465)

[Resultado: 26](#_Toc163378466)

[**11.** **Triggers** 27](#_Toc163378467)

[*Trigger 1:* 27](#_Toc163378468)

[Objetivo: 27](#_Toc163378469)

[SQL: 27](#_Toc163378470)

[Resultado: 28](#_Toc163378471)

[*Trigger 2:* 28](#_Toc163378472)

[*Objetivo:* 28](#_Toc163378473)

[SQL: 28](#_Toc163378474)

[Resultado: 29](#_Toc163378475)

[**12.** **Conclusión** 29](#_Toc163378476)

# **Introducción**

## *Versión 0.1*

Este es el comienzo de mi proyecto de base de datos, en el podremos ver cómo va avanzando el curso. Con el paso del tiempo esta introducción se quedará corta, pero esa es la finalidad de este proyecto, ver como todo lo que hemos ido aprendiendo en cada semana del curso queda plasmado no solo en exámenes si no también en algo más personal.

El objetivo de este será formar una Base de datos personalizada sobre los eSports en la que podamos poner en práctica lo visto en las clases por lo tanto buscaremos hacer una base con recursivas, herencias, entidades débiles y las relaciones necesarias para dar sentido y forma a nuestro “pequeño”.

Y como ya podremos comprobar esta introducción se irá ampliando a lo largo de los meses añadiendo lo que trabajaremos en ella con cada versión.

## *Versión 0.2*

Una vez tengamos hecha la descripción haremos su modelo E/R y haremos el paso a tablas en DBeaver para en la siguiente parte del proyecto cargar de datos toda la base de datos y darle un sentido a esta.

## *Version 0.3*

Ahora que ya tenemos todas las tablas comenzaremos con el poblado masivo de estas, para hacer esto usaremos tanto [Google Datasheet](https://docs.google.com/spreadsheets/create?hl=es) para poder modificar nuestras tablas a gusto e insertar los valores manualmente en caso de ser necesario, como [esta página](https://www.onlinedatagenerator.com/home/demo) para generar masivamente los datos que puedan ser generados de forma automatizada.

## *Version 0.4*

Con todos los datos ya cargados nos dispondremos a realizar las 5 consultas necesarias para el proyecto, estas 5 consultas serán generadas mediante un script SQL el cual adjuntaré en su respectivo [apartado](#_Consultas_SQL), junto con la imagen con el resultado.

# **Descripción**

En el proyecto como ya hemos dicho en la [introducción](#_Versión_0.1) haremos la base de datos sobre los eSports y concretamente del videojuego Counter Strike Global Offensive, este tiene múltiples eventos a lo largo del año en el que participan múltiples equipos de las diferentes ligas del mundo.

Una vez sepamos esto necesitaremos concretar los datos necesarios para la base de datos, para ello necesitaremos saber el nombre del torneo y los diferentes tipos torneos que se celebrarán, además serán de 3 tipos, estos pueden ser mundiales, continentales o nacionales. También cuentan con diferentes condiciones de victoria, estas serán el mejor de 1, el mejor de 3 y el mejor de 5, necesitamos saber el tipo de condición de la victoria de cada torneo.

Necesitaremos saber la fecha en que se celebra cada torneo, en la que acaba y en qué región se celebra. Algunos torneos son clasificatorios de otros, por ejemplo, el ESL Challenger League es un torneo que se realiza en Norteamérica, Europa, Asia y el Pacífico y los 2 primeros de cada una de estas ubicaciones tienen clasificación directa a la ESL Pro League.

De cada región necesitaremos almacenar un código único el cual será una abreviatura de su nombre y el nombre completo.

De las sesiones necesitamos almacenar su fecha y su hora de inicio y fin, además necesitaremos saber que ronda se juega en la sesión pudiendo ser Octavos, Cuartos, Semifinales y Finales. Necesitaremos saber en qué estadio se celebra la sesión y a que torneo pertenece esa sesión sabiendo que no se podrá celebrar una sesión en un mismo estadio durante un mismo torneo.

En los partidos se necesitará almacenar el identificador único de cada partido, la hora en que se celebra el partido junto con la que acaba.

Cada partido se celebra entre 2 equipos, necesitaremos saber el resultado de ambos en equipos en cada partido.

En cada equipo se deseará almacenar un identificador único para cada equipo, su nombre completo, su abreviatura, un año de fundación y un CEO o representante del equipo.

De cada jugador se deseará almacenar su DNI, nombre y apellidos, su fecha de nacimiento, su nacionalidad y su edad.

Necesitaremos saber a qué equipo pertenece cada jugador, cada jugador solo podrá pertenecer a un equipo y un equipo solo está formado por 5 jugadores.

Además, las sesiones tienen lugar en diferentes estadios donde se necesitarán almacenar sus identificadores únicos, junto con su país y ciudad de ubicación, además del aforo que tienen.

Para cada sesión habrá entradas, en las entradas necesitaremos saber su identificador único, su precio, si es online o presencial y el Stock. Estas entradas van asociadas a una sola sesión, permitiendo ver todos los partidos que se celebren en esa sesión, pero no los de las demás sesiones del torneo.

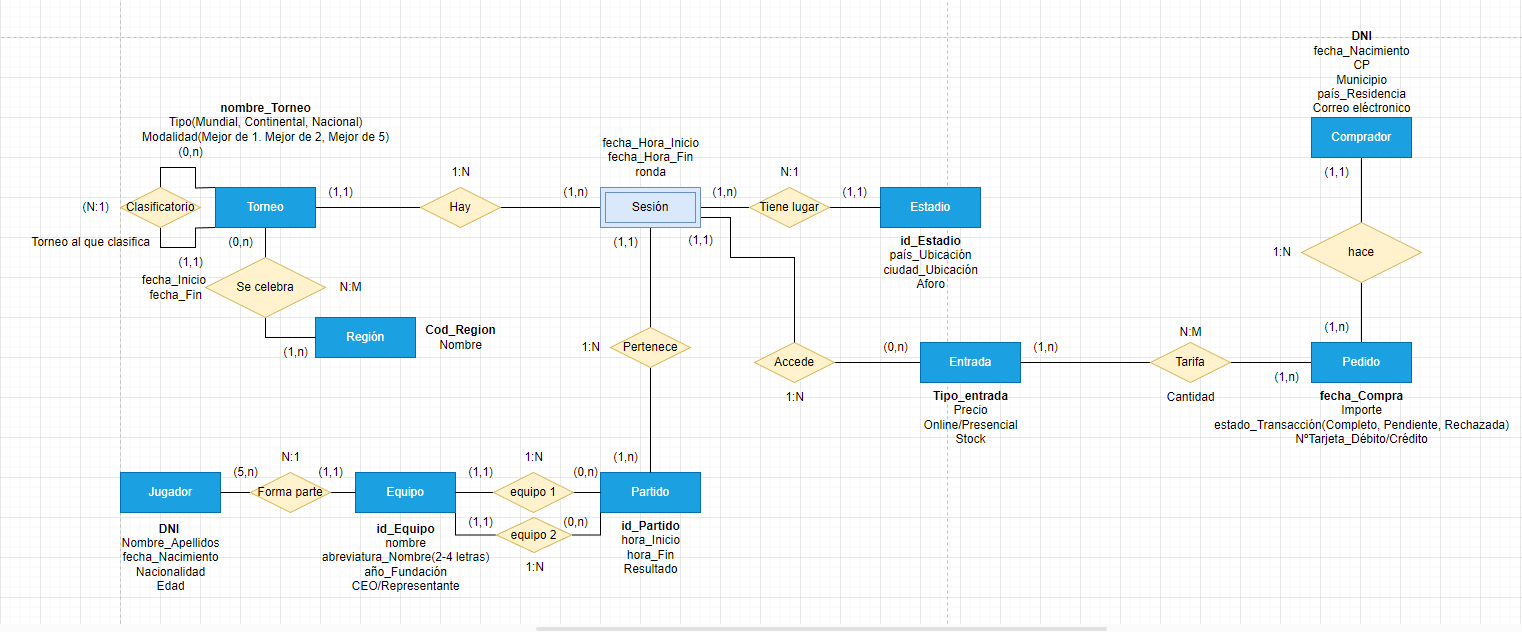
También necesitaremos guardar los pedidos, de cada pedido almacenaremos su fecha de compra, el importe, el estado de la transacción pudiendo ser completo pendiente o rechazado y el nº de tarjeta.

Los clientes podrán realizar varios pedidos que irán asociados a la fecha de compra. Además de cada pedido necesitaremos saber la cantidad de entradas que se van a pagar en el pedido.

Por último, necesitaremos saber de los compradores o asistentes del torneo su DNI, su fecha de nacimiento, su correo electrónico, su código postal, su municipio, su país de residencia y su correo electrónico.

# **Modelo Entidad/Relación extendido y Modelo relacional.**

## MER.



## Modelo relacional.

Región (**Cod\_Región**, Nombre)

PK: Cod\_Región(Región)

Torneo (**nombre\_Torneo**, Tipo, Modalidad, Es\_clasificatorio)

PK: nombre\_Torneo(Torneo)

Torneo\_Se\_Celebra\_En\_Región (**Nombre\_Torneo**(Torneo), **Cod\_Región**(Región), fecha\_Inicio, fecha\_Fin)

PK: Nombre\_Torneo(Torneo), Cod\_Región(Región)

FK: Nombre\_Torneo(Torneo)

FK: Cod\_Región(Región)

Sesión (fecha\_Hora\_Inicio, fecha\_Hora\_Fin, ronda, **nombre\_Torneo**(Torneo), **id\_Estadio**(Estadio))

PK: id\_Estadio(Estadio), nombre\_Torneo(Torneo)

FK: nombre\_Torneo(Torneo)

FK: id\_Estadio(Estadio).

Estadio (**id\_Estadio**, país\_Ubicación, ciudad\_Ubicación, Aforo)

PK: id\_Estadio(Estadio)

Partido (**id\_Partido**, hora\_Inicio, hora\_Fin, Resultado, id\_Estadio(Sesión), Equipo1(Equipo), Equipo2(Equipo))

PK: id\_Partido(Partido)

FK: id\_Estadio(Sesión)

FK: Equipo1(Equipo)

FK: Equipo2(Equipo).

Equipo (**id\_Equipo**, nombre, abreviatura\_Nombre, año\_Fundación, CEO/Representante)

PK: id\_Equipo(Equipo)

Jugador (**DNI**, Nombre\_Apellidos, fecha\_Nacimiento, Edad, Nacionalidad, id\_Equipo(Equipo))

PK: DNI (Jugador).

FK: id\_Equipo (Equipo).

Entrada (**Tipo\_Entrada**, Precio, Online/Presencial, Stock, **id\_Estadio**(Sesión))

PK: Tipo\_Entrada(Entrada)

FK: id\_Estadio(Sesión)

Tarifa (**Tipo\_Entrada**(Entrada), **fecha\_Compra**(Pedido), Cantidad)

PK: Tipo\_Entrada(Entrada), fecha\_Compra(Pedido)

FK: id\_Entrada(Entrada)

FK: fecha\_Compra(Pedido)

Pedido (**fecha\_Compra**, Importe, estado\_Transacción, NºTarjeta\_Débito/Crédito, DNI(Comprador))

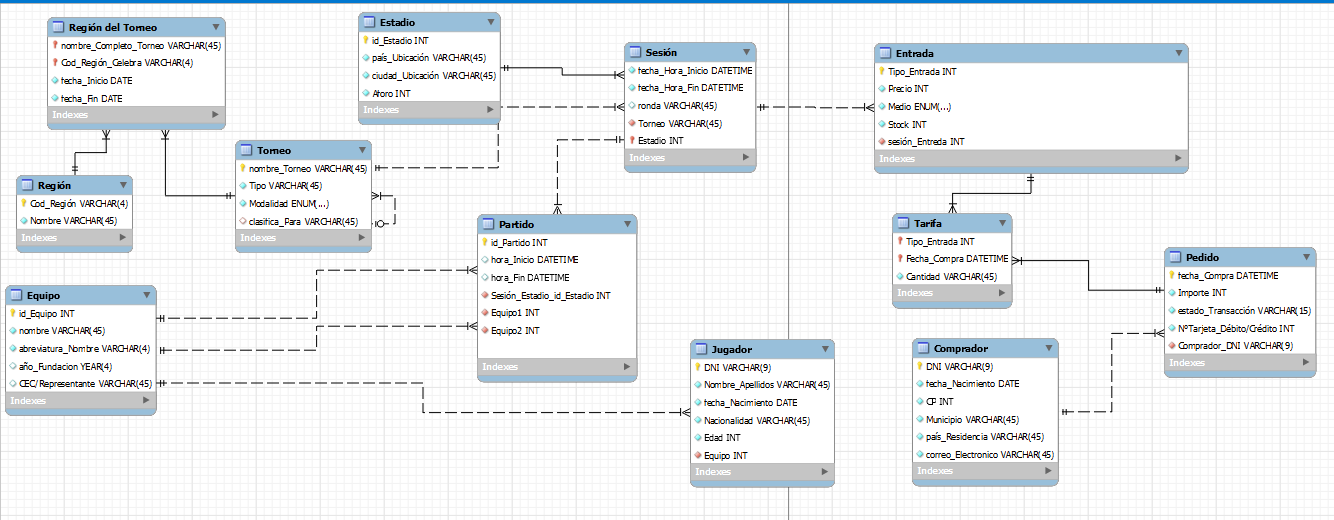
PK: fecha\_Compra(Pedido)

FK: DNI(Comprador)

Comprador (**DNI**, fecha\_Nacimiento, CP, Municipio, país\_Residencia, correo\_Electronico)

PK: DNI(Comprador).

# **Paso a tablas en MySQL Workbench.**



# **Importado a MySQL**

Adjunto el archivo con el [Sql de la base de datos](https://github.com/magugolenc/Proyecto-Base-de-datos/blob/main/ProyectoSQL%20JGG).

Adjunto el archivo con el [Sql de la base de datos](https://github.com/magugolenc/Proyecto-Base-de-datos/blob/main/TorneoCsgoCopiaCompletaSQL.sql) y la carga de datos realizada.

# **Actualizaciones y modificaciones**

## *Modelo relacional:*

Torneo (**nombre\_Torneo**, Tipo, Modalidad,Premio, clasifica\_Para)

PK: nombre\_Torneo(Torneo)

FK: Nombre\_Torneo(Torneo)

Estadio (**id\_Estadio**, país\_Ubicación, provincia\_Ubicación, ciudad\_Ubicación, Aforo)

PK: id\_Estadio(Estadio)

Partido (**id\_Partido**, hora\_Inicio, hora\_Fin, ResultadoEquipo1, ResultadoEquipo2, Sesion\_Estadio\_id\_Estadio(Sesión), Equipo1(Equipo), Equipo2(Equipo))

PK: id\_Partido(Partido)

FK: id\_Estadio(Sesión)

FK: Equipo1(Equipo)

FK: Equipo2(Equipo).

Jugador (**DNI**, Nombre\_Jugador, fecha\_Nacimiento, Edad, Nacionalidad, id\_Equipo(Equipo))

PK: DNI (Jugador).

FK: id\_Equipo (Equipo).

Tarifa (**Tipo\_Entrada**(Entrada), **fecha\_Compra**(Pedido), Cantidad, Estadio\_Sesion(Entrada)

PK: Tipo\_Entrada(Entrada), fecha\_Compra(Pedido)

FK: id\_Entrada(Entrada)

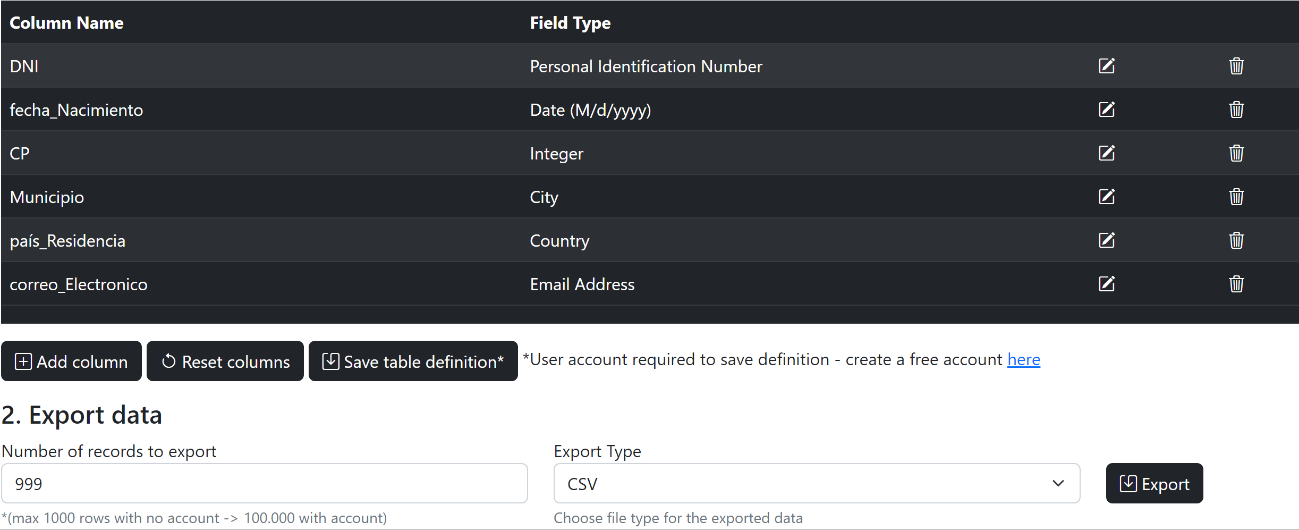
FK: fecha\_Compra(Pedido)

FK: Estadio\_Sesion(Entrada)

# **Carga de datos masiva**

Para realizar la carga de datos masiva de nuestra base de datos usaremos la página web [onlinedatagenerator](https://www.onlinedatagenerator.com). Esta página dedicada a la carga masiva de datos nos permite configurar muchos apartados, así como el del identificador numérico personal, correos electrónicos, fechas en diferentes formatos y demás opciones. Tiene un pequeño problema y es el código postal con el cual no cuenta de forma predeterminada, aunque como esto es un ejemplo de base de datos y no una real, usaremos el formato integer del 1 al 99999.

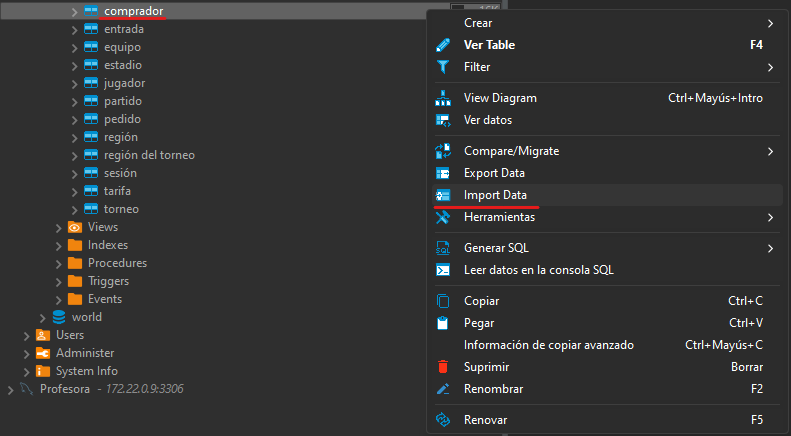
Como bien podemos ver en esta foto aquí tenemos seleccionada toda la configuración necesaria.



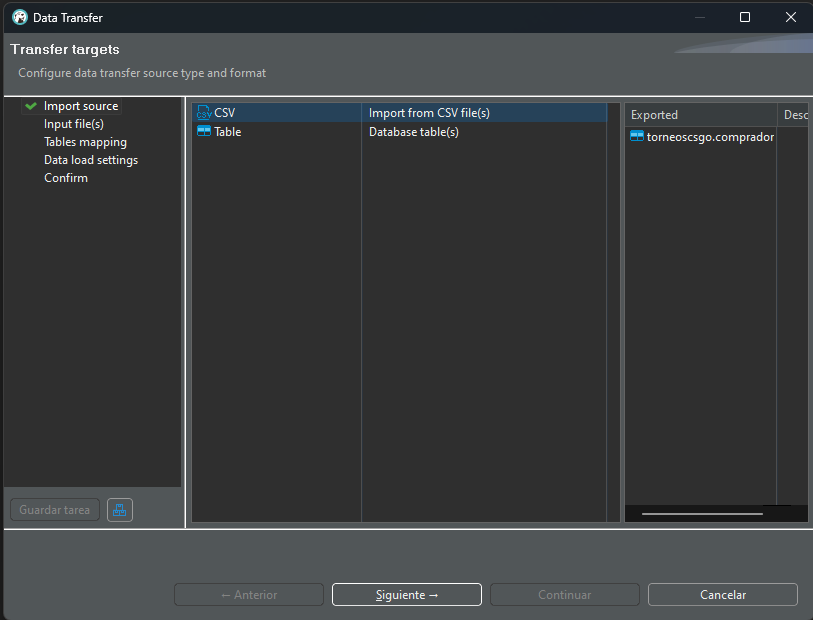
Además, como he dicho antes hemos podido limitar el integer, pues eso mismo haremos con la fecha de nacimiento la cual está establecida entre el 1/1/1950 y el 1/1/2005 esto lo haremos para evitar posibles compradores menores de edad y evidentemente que un bebe de 0 años pueda “comprar” una entrada. También podremos seleccionar el formato en el que exportará el archivo generado y el número de filas de datos que tendrá nuestra tabla, en este caso la de compradores.

## **Tutorial de importación de csv para la carga masiva de datos.**

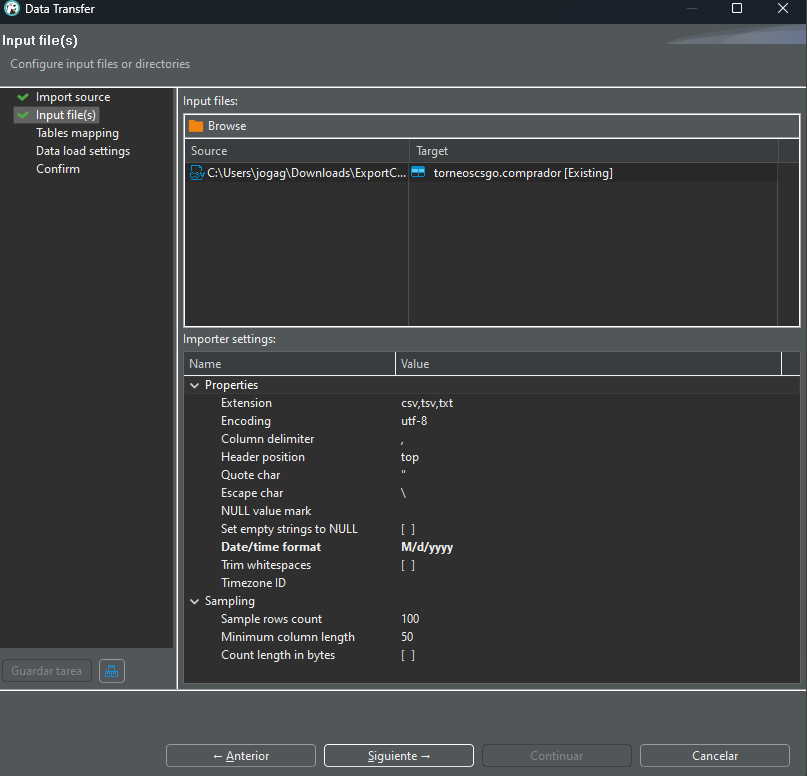
Una vez generado el csv con los datos, le daremos clic derecho en la tabla de comprador (en la que queremos insertar los datos), nos desplegara una serie de opciones y seleccionaremos “Import Data”.



Seleccionaremos el formato para importarlo en csv que es el que anteriormente escogimos en el onlinedatagenerator.

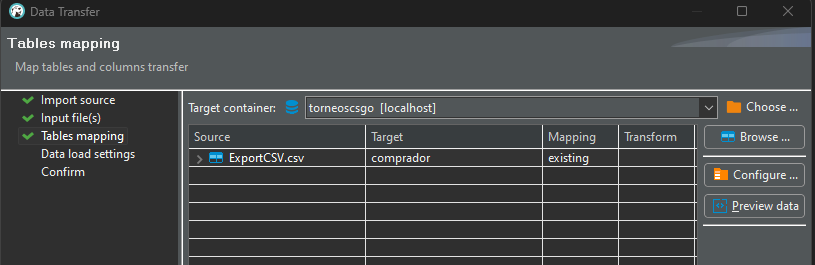


Al darle a siguiente nos abrirá el explorador de archivos donde seleccionaremos nuestro csv y nos saldrá lo siguiente:

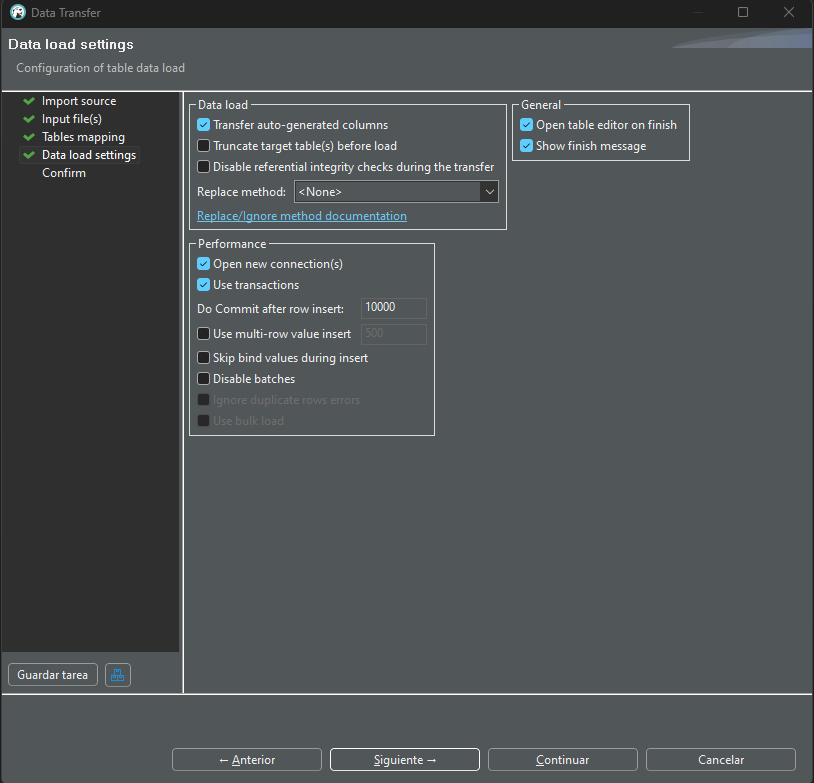


Aquí tendremos que configurar lo que nos pide, en mi caso solo cambio la fecha la cual modifico el formato a como se ha hecho en el csv, aunque en las próximas tablas que poblaremos necesitaremos activar el apartado de set empty strings to NULL ya que en esta tenemos configurado que no pueden ser valores nulos o vacíos, pero en otras si que pueden aparecer estos valores NULL, en ese caso activaremos esa opción.

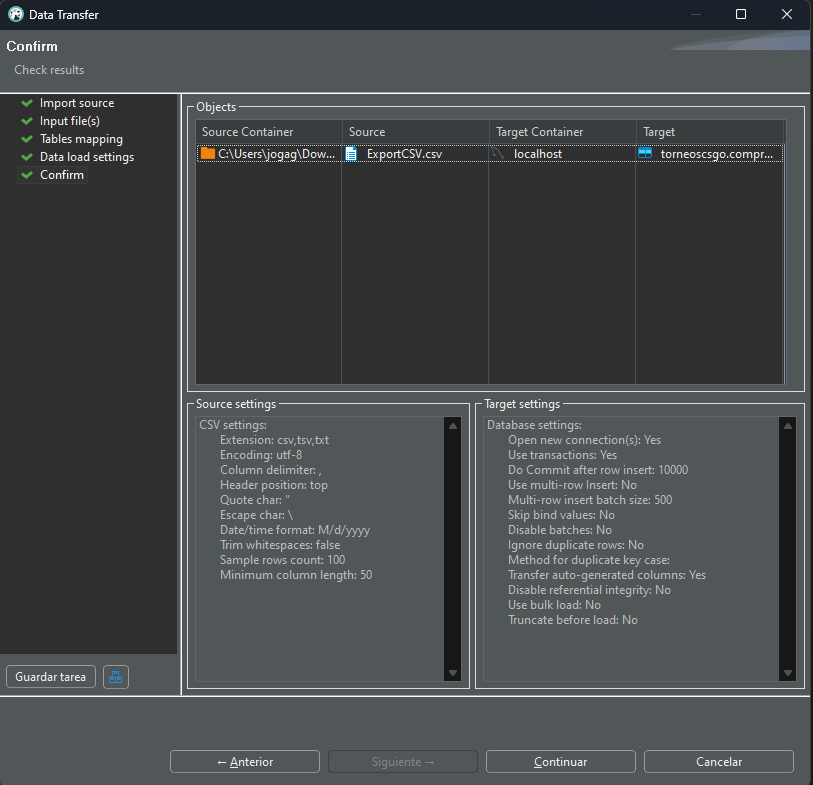
Una vez configurado seleccionaremos la base de datos en la que queramos añadirla y en este caso al ya haberla seleccionado anteriormente con el clic derecho no hará falta, pero en caso de no hacerlo así, en target pondremos la tabla y en Mapping al ya tener creada la tabla existing.



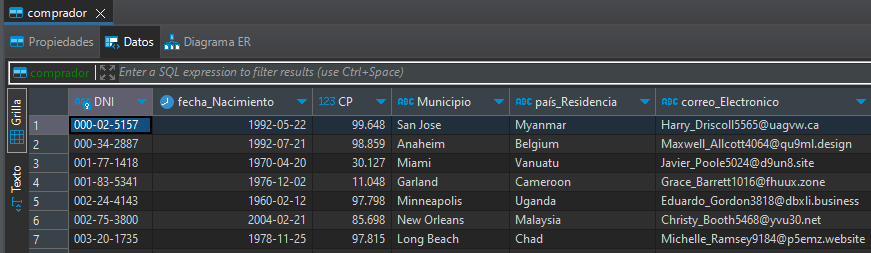
Este apartado no será necesario cambiarlo, lo dejaremos por defecto.



Por último, confirmaremos el importado y ya estará listo si no tenemos formatos y datos erróneos.



Resultado:



Continuaremos poblando las tablas hasta haber poblado todas.

# **Consultas SQL**

## *Objetivo:*

Necesitaremos saber qué equipos son los que han tenido un mejor rendimiento a nivel Nacional, buscando así qué equipos han ganado más encuentros en sus respectivos campeonatos Nacionales.

## *SLQ:*

**use** torneoscsgo;

**select** **concat**(**"ID: "**, *e*.id\_Equipo,**" Nombre: "**, *e*.nombre) **AS** **'InfoEquipo'**, **COUNT**(*p*.id\_Partido) **AS** **'Victorias\_Nacionales'** **from** equipo *e*

**inner** **join** partido *p* **ON** *e*.id\_Equipo = *p*.Equipo1 **OR** *e*.id\_Equipo = *p*.Equipo2

**inner** **join** sesión s **ON** *p*.Sesion\_Estadio\_id\_Estadio = ***s***.Estadio

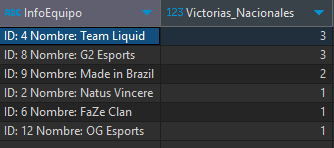
**inner** **join** torneo *t* **ON** s.Torneo = t.nombre\_Torneo

**where** (*p*.Equipo1 = *e*.id\_Equipo **AND** *p*.ResultadoEquipo1 > *p*.ResultadoEquipo2) **or** (*p*.Equipo2 = *e*.id\_Equipo **AND** *p*.ResultadoEquipo2 > *p*.ResultadoEquipo1) **and** *t*.Tipo = **'Nacional'**

**group** **by** *e*.id\_Equipo

**order** **by** **count**(*p*.id\_Partido) **DESC**;

## *Resultado:*



## *Objetivo:*

Ahora queremos saber que equipos cuentan con una plantilla de jóvenes promesas, por lo cual necesitaremos saber los equipos que con jugadores menores de 20 años y adicionalmente buscaremos los que necesitan renovar plantilla con lo que buscaremos también a los que tienen jugadores mayores de 28 años.

## *SLQ 2:*

**select** **concat**(**"Menores ID: "**, *e*.id\_Equipo,**" Nombre: "**, *e*.nombre) **as** **'InfoEquipo'**, **count**(*j*.Edad<20) **as** **'NumeroDeJugadores'** **from** equipo *e*

**inner** **join** jugador *j* **on** *e*.id\_Equipo = *j*.Equipo

**where** *j*.Edad < 20

**group** **by** *e*.id\_Equipo

**union**

**select** **concat**(**"Mayores ID: "**, *e*.id\_Equipo,**" Nombre: "**, *e*.nombre) **as** **'InfoEquipo'**, **count**(*j*.Edad>28) **as** **'NumeroDeJugadores'from** equipo *e*

**inner** **join** jugador *j* **on** *e*.id\_Equipo = *j*.Equipo

**where** *j*.Edad > 28

**group** **by** *e*.id\_Equipo;

## *Resultado:*



## *Objetivo:*

Necesitamos saber los partidos de los que se han llevado a cabo han tenido que jugar más de 20 rondas para finalizar el partido, además del nombre del equipo y la fecha en la que se celebró. Tendremos que agruparlos por la fecha ordenándolos por fecha de mayor a menor.

## *SLQ 3:*

**select** **concat**(*p*.hora\_Inicio, **"<>"**, *p*.hora\_Fin) **as** **'HorarioPartido'**, *e*.nombre **as** **'NombreEquipo1'**, *e2*.nombre **as** **'NombreEquipo2'**, **sum**(*p*.ResultadoEquipo1+*p*.ResultadoEquipo2) **as** **'TotalDeRondas'**

**from** partido *p*

**inner** **join** equipo *e* **on** *p*.Equipo1 = *e*.id\_Equipo

**inner** **join** equipo *e2* **on** *p*.Equipo2 = *e2*.id\_Equipo

**group** **by** **concat**(*p*.hora\_Inicio, **"<>"**, *p*.hora\_Fin), *e*.nombre, *e2*.nombre

**having** **sum**(*p*.ResultadoEquipo1+*p*.ResultadoEquipo2)>20

**order** **by** ***HorarioPartido*** **desc**;

## *Resultado:*



## *Objetivo:*

En la cuarta consulta buscaremos saber que meses son los que han dado mejor rendimiento económico en todo el tiempo de ventas y que meses han sido los que más gente ha realizado un pedido. Para esto sacaremos los meses, el total de dinero generado de cada mes, y el total de pedidos realizados en cada mes.

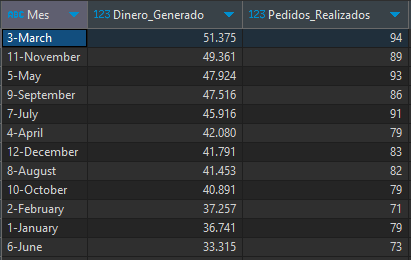
## *SLQ 4:*

**select** **concat**(**month**(*p*.fecha\_Compra),**'-'**, **monthname**(*p*.fecha\_Compra)) **as** *Mes*, **sum**(*p*.Importe) **as** *Dinero\_Generado*, **count**(\*) **as** *Pedidos\_Realizados* **from** pedido *p*

**group** **by** *Mes*

**order** **by** *Dinero\_Generado* **desc**;

## *Resultado:*



## *Objetivo:*

Necesitaremos saber los países desde los que se han comprado las entradas para asistir a los partidos número 5 y 10 para un estudio de mercado, sacaremos el nombre del país la cantidad de gente proveniente de ese país. Ordenaremos por el número de ventas totales de cada país.

## *SLQ 5:*

**SELECT** *c*.***pa***ís\_Residencia **AS** País, **COUNT**(\*) **AS** *Cantidad\_Ventas*

**FROM** pedido *p*

**JOIN** comprador *c* **ON** *p*.Comprador\_DNI = *c*.DNI

**JOIN** entrada *e* **ON** *p*.tipo\_entrada = *e*.Tipo\_Entrada **AND** *p*.estadio\_sesion = *e*.Estadio\_Sesion

**JOIN** partido *pt* **ON** *e*.Estadio\_Sesion = *pt*.Sesion\_Estadio\_id\_Estadio

**WHERE** *pt*.id\_Partido **IN** (7, 8)

**GROUP** **BY** *c*.***pa***ís\_Residencia

**ORDER** **BY** Cantidad\_Ventas **DESC**;

## *Resultado:*



# **Vistas**

## *Vista de la* [*consulta nº 1.*](#_Objetivo:)

### SQL:

**CREATE** **VIEW** Victorias\_de\_cada\_equipo **AS** **select** **concat**(**"ID: "**, e.id\_Equipo,**" Nombre: "**, e.nombre) **AS** **'InfoEquipo'**, **COUNT**(p.id\_Partido) **AS** **'Victorias\_Nacionales'** **from** equipo e

**inner** **join** partido p **ON** e.id\_Equipo = p.Equipo1 **OR** e.id\_Equipo = p.Equipo2

**inner** **join** sesión s **ON** p.Sesion\_Estadio\_id\_Estadio = s.Estadio

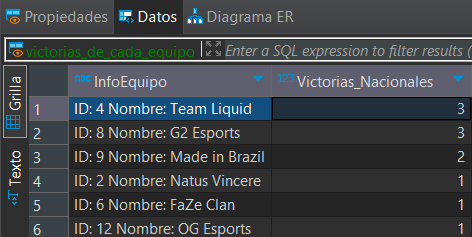
**inner** **join** torneo t **ON** s.Torneo = t.nombre\_Torneo

**where** (p.Equipo1 = e.id\_Equipo **AND** p.ResultadoEquipo1 > p.ResultadoEquipo2) **or** (p.Equipo2 = e.id\_Equipo **AND** p.ResultadoEquipo2 > p.ResultadoEquipo1) **and** t.Tipo = **'Nacional'**

**group** **by** e.id\_Equipo

**order** **by** **count**(p.id\_Partido) **DESC**;

### Resultado:



## *Vista de la* [*consulta nº 4.*](#_Objetivo:_1)

### SQL:

**CREATE** **VIEW** Ingresos\_y\_pedidos\_totales\_por\_mes **AS**

**select** **concat**(**month**(p.fecha\_Compra),**'-'**, **monthname**(p.fecha\_Compra)) **as** Mes, **sum**(p.Importe) **as** Dinero\_Generado, **count**(\*) **as** Pedidos\_Realizados **from** pedido p

**group** **by** Mes

**order** **by** Dinero\_Generado **desc**;

### Resultado:



# **Funciones y procedimientos**

## *Función 1:*

### Objetivo:

Calcular los ingresos totales de un torneo en base a los pedidos realizados para dicho torneo.

### SQL:

**DELIMITER $$**

**CREATE** **FUNCTION** calcularIngresosTorneo(Nombre\_torneo **VARCHAR**(45)) **RETURNS** **DECIMAL**(10,2)

**DETERMINISTIC**

**BEGIN**

**DECLARE** total\_income **DECIMAL**(10,2);

**SELECT** **SUM**(Importe) **INTO** total\_income

**FROM** pedido

**WHERE** tipo\_entrada **IN** (

**SELECT** Tipo\_Entrada

**FROM** entrada

**WHERE** Estadio\_Sesion **IN** (

**SELECT** Estadio

**FROM** sesión

**WHERE** Torneo = Nombre\_torneo

)

);

**RETURN** total\_income;

**END** $$

**DELIMITER ;**

### Resultado:

**SELECT** calcularIngresosTorneo(**'eSL Liga'**);



## *Función 2:*

### Objetivo:

Calcular el porcentaje de victorias de un equipo en un torneo específico.

### SQL:

**DELIMITER $$**

**CREATE** **FUNCTION** calcular\_porcentaje\_victorias\_equipo\_en\_torneo(id\_equipo **INT**, nombre\_torneo **VARCHAR**(45)) **RETURNS** **DECIMAL**(5, 2)

**DETERMINISTIC**

**BEGIN**

**DECLARE** total\_partidos **INT**;

**DECLARE** total\_victorias **INT**;

**DECLARE** porcentaje\_victorias **DECIMAL**(5, 2);

**SELECT** **COUNT**(\*) **INTO** total\_partidos

**FROM** partido p

**JOIN** sesión s **ON** p.Sesion\_Estadio\_id\_Estadio = s.Estadio

**JOIN** torneo t **ON** s.Torneo = t.nombre\_Torneo

**WHERE** (p.Equipo1 = id\_equipo **OR** p.Equipo2 = id\_equipo) **AND** t.nombre\_Torneo = nombre\_torneo;

**SELECT** **SUM**(**CASE** **WHEN** p.ResultadoEquipo1 > p.ResultadoEquipo2 **AND** p.Equipo1 = id\_equipo **THEN** 1

**WHEN** p.ResultadoEquipo2 > p.ResultadoEquipo1 **AND** p.Equipo2 = id\_equipo **THEN** 1 **ELSE** 0 **END**) **INTO** total\_victorias

**FROM** partido p

**JOIN** sesión s **ON** p.Sesion\_Estadio\_id\_Estadio = s.Estadio

**JOIN** torneo t **ON** s.Torneo = t.nombre\_Torneo

**WHERE** (p.Equipo1 = id\_equipo **OR** p.Equipo2 = id\_equipo) **AND** t.nombre\_Torneo = nombre\_torneo;

**IF** total\_partidos > 0 **THEN**

**SET** porcentaje\_victorias = (total\_victorias / total\_partidos) \* 100;

**ELSE**

**SET** porcentaje\_victorias = 0;

**END** **IF**;

**RETURN** porcentaje\_victorias;

**END** $$

**DELIMITER ;**

### Resultado:

**SELECT** calcular\_porcentaje\_victorias\_equipo\_en\_torneo(**'6'**,**'eSL Liga'**);



## *Procedimiento 1:*

### Objetivo:

Gestionar la información de un equipo, actualizando su abreviatura si el equipo ya existe en la base de datos o creando un nuevo equipo con una abreviatura si no existe.

### SQL:

**DROP** **PROCEDURE** **IF** **EXISTS** gestionar\_informacion\_equipo;

**DELIMITER $$**

**CREATE** **PROCEDURE** gestionar\_informacion\_equipo(**IN** nombre\_equipo **VARCHAR**(45), **IN** nueva\_abreviatura **VARCHAR**(4))

**BEGIN**

**DECLARE** equipo\_existente **INT**;

**DECLARE** nuevo\_id\_equipo **INT**;

**SELECT** **COUNT**(\*) **INTO** equipo\_existente **FROM** equipo **WHERE** nombre = nombre\_equipo;

**IF** equipo\_existente > 0 **THEN**

**UPDATE** equipo **SET** abreviatura\_Nombre = nueva\_abreviatura **WHERE** nombre = nombre\_equipo;

**SELECT** **CONCAT**(**'Se ha actualizado la abreviatura del equipo '**, ***nombre\_equipo***, **' a '**, ***nueva\_abreviatura***) **AS** *Mensaje*;

**ELSE**

**SELECT** (**MAX**(id\_Equipo)+1) **INTO** nuevo\_id\_equipo **FROM** equipo;

**SET** nuevo\_id\_equipo = **IFNULL**(nuevo\_id\_equipo, 0) + 1;

**INSERT** **INTO** equipo (id\_Equipo, nombre, abreviatura\_Nombre) **VALUES** (***nuevo\_id\_equipo***, ***nombre\_equipo***, ***nueva\_abreviatura***);

**SELECT** **CONCAT**(**'Se ha creado un nuevo equipo con ID '**, ***nuevo\_id\_equipo***, **', nombre '**, ***nombre\_equipo***, **' y abreviatura '**, ***nueva\_abreviatura***) **AS** *Mensaje*;

**END** **IF**;

**END** $$

**DELIMITER ;**

### Resultado:

**CALL** gestionar\_informacion\_equipo(**'Prueba nuevo equipo'**,**'PNE'**);



**CALL** gestionar\_informacion\_equipo(**'Prueba nuevo equipo'**,**'PNN'**);



## *Procedimiento 2:*

### Objetivo:

Generar un informe de los ingresos totales y el número total de pedidos realizados para un torneo específico.

### SQL:

**DELIMITER $$**

**CREATE** **PROCEDURE** generar\_reporte\_ingresos\_torneo(**IN** nombre\_torneo **VARCHAR**(45))

**BEGIN**

**DECLARE** total\_ingresos **DECIMAL**(10, 2);

**DECLARE** total\_pedidos **INT**;

**SET** total\_ingresos = calcularIngresosTorneo(nombre\_torneo);

**SELECT** **COUNT**(\*) **INTO** total\_pedidos

**FROM** pedido

**WHERE** tipo\_entrada **IN** (

**SELECT** Tipo\_Entrada

**FROM** entrada

**WHERE** Estadio\_Sesion **IN** (

**SELECT** Estadio

**FROM** sesión

**WHERE** Torneo = nombre\_torneo

)

);

**SELECT** **CONCAT**(**'Los ingresos totales del torneo '**, nombre\_torneo, **' son: '**, total\_ingresos,**'€.'**) **AS** Total\_Ingresos,

**CONCAT**(**'El número total de pedidos realizados en el torneo '**, nombre\_torneo, **' es: '**, total\_pedidos) **AS** Total\_Pedidos;

**END** $$

**DELIMITER ;**

### Resultado:

**CALL** generar\_reporte\_ingresos\_torneo(**'eSL Liga'**);



## *Procedimiento 3:*

### Objetivo:

Generar un informe de los DNI de los jugadores pertenecientes a un equipo específico.

### SQL:

**DELIMITER $$**

**CREATE** **PROCEDURE** generar\_informe\_DNI\_jugadores\_equipo(**IN** nombre\_equipo **VARCHAR**(45))

**BEGIN**

**DECLARE** done **BOOLEAN** **DEFAULT** **FALSE**;

**DECLARE** dni\_jugador **VARCHAR**(11);

**DECLARE** informacion\_jugadores **VARCHAR**(2000) **DEFAULT** **''**;

**DECLARE** tabla\_jugadores **VARCHAR**(2000) **DEFAULT** **''**;

**DECLARE** jugador\_cursor **CURSOR** **FOR**

**SELECT** DNI **FROM** jugador

**WHERE** Equipo = (

**SELECT** id\_Equipo **FROM** equipo

**WHERE** nombre = nombre\_equipo

);

**DECLARE** **CONTINUE** HANDLER **FOR** **NOT** **FOUND** **SET** done = **TRUE**;

**OPEN** jugador\_cursor;

read\_loop: **LOOP**

**FETCH** jugador\_cursor **INTO** dni\_jugador;

**IF** done **THEN**

**LEAVE** read\_loop;

**END** **IF**;

**SET** informacion\_jugadores = **CONCAT\_WS**(**' '**,dni\_jugador);

**SET** tabla\_jugadores = **CONCAT**(tabla\_jugadores, informacion\_jugadores, **'\n'**);

**END** **LOOP**;

**CLOSE** jugador\_cursor;

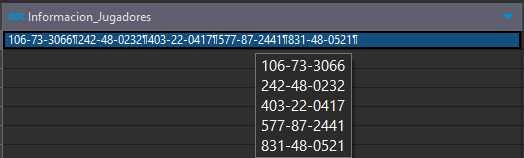
**SELECT** ***tabla\_jugadores*** **AS** *Informacion\_Jugadores*;

**END** $$

**DELIMITER ;**

### Resultado:

**CALL** generar\_informe\_jugadores\_equipo(**'G2 Esports'**);



# **Triggers**

## *Trigger 1:*

### Objetivo:

Guardar los cambios en el estado de los pedidos y registrarlos en una tabla de auditoría.

### SQL:

**CREATE** **TABLE** auditoria\_pedidos(

fecha\_compra **DATETIME**,

accion **VARCHAR**(100),

fecha\_modificacion **DATETIME**,

**PRIMARY** **KEY** (fecha\_compra),

**FOREIGN** **KEY** (fecha\_compra) **REFERENCES** pedido(fecha\_Compra)

);

**DROP** **TRIGGER** **IF** **EXISTS** audit\_pedidos;

**DELIMITER $$**

**CREATE** **TRIGGER** audit\_pedidos

**AFTER** **UPDATE** **ON** pedido

**FOR** **EACH** **ROW**

**BEGIN**

**DECLARE** accion **VARCHAR**(100) **DEFAULT** **''**;

**IF** **OLD**.estado\_Transacción <> **NEW**.estado\_Transacción **THEN**

**SET** accion = **CONCAT**(**'Estado cambiado de '**, **OLD**.estado\_Transacción, **' a '**, **NEW**.estado\_Transacción);

**INSERT** **INTO** ***auditoria\_pedidos*** (***fecha\_compra*** ,***accion***, ***fecha\_modificacion***)

**VALUES** (**OLD**.fecha\_Compra , ***accion***, **NOW**());

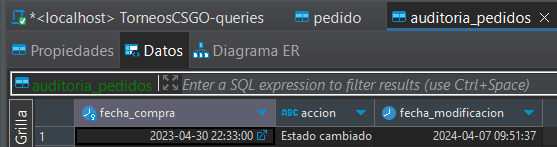
**END** **IF**;

**END** $$

**DELIMITER ;**

### Resultado:





## *Trigger 2:*

## *Objetivo:*

Actualizar el stock de entradas en base a los pedidos realizados, excepto si la entrada es de tipo online.

### SQL:

**DELIMITER $$**

**CREATE** **TRIGGER** actualizar\_stock

**AFTER** **INSERT** **ON** pedido

**FOR** **EACH** **ROW**

**BEGIN**

**DECLARE** cantidad\_comprada **INT**;

**DECLARE** tipo\_entrada\_online **ENUM**(**'online'**, **'presencial'**);

**SET** cantidad\_comprada = **NEW**.cantidad;

**SET** tipo\_entrada\_online = (**SELECT** Medio **FROM** entrada **WHERE** Tipo\_Entrada = **NEW**.tipo\_entrada **AND** Estadio\_Sesion = **NEW**.estadio\_sesion);

**IF** tipo\_entrada\_online != **'online'** **THEN**

**UPDATE** entrada

**SET** Stock = Stock - cantidad\_comprada

**WHERE** Tipo\_Entrada = **NEW**.tipo\_entrada **AND** Estadio\_Sesion = **NEW**.estadio\_sesion;

**END** **IF**;

**END** $$

**DELIMITER ;**

### Resultado:





# **Conclusión**

En este proyecto hemos desarrollado a nuestra base de datos como si de nuestro niño pequeño se tratase lo hemos ido mimando desde su nacimiento, con el desarrollo del modelo entidad relación extendido, hasta su inserción masiva de datos. Con el hemos reforzado nuestro criterio en cuanto a la facilidad de uso de una base datos además de mejorar el entendimiento de estas.

Además, hemos aprendido a usar más a fondo DBeaver ya que no todo es SQL, también es importante el entorno de trabajo que usamos para nuestra base de datos, DBeaver en mi caso. Aprendiendo así a realizar todo lo que tiene que ver con el DDL en un entorno gráfico mucho más sencillo y práctico.

No obstante, también he tenido varias situaciones de incertidumbre ya que, al ser una nueva herramienta y un ámbito totalmente nuevo, he necesitado ir madurando y mejorando por el camino mis habilidades con las bases de datos, sobre todo con el SQL.

Por último, quería resaltar el gran trabajo de originalidad y la capacidad de desarrollar algo nuevo a partir de un tema en concreto como puede ser generar una base de datos completa en base a un videojuego, plantear que preguntas quieres hacerle a la base de datos para sacar información con sentido para los fines estadísticos o de curiosidad para algunas personas, viendo así las capacidades de cada uno siendo un reto no solo en el ámbito de las bases de datos, sino que también en muchos otros.

Gracias a lo visto en clase y a los diferentes triggers, funciones, etc, podemos ver información estadística de nuestra base de datos además de ahorrarnos tiempo en cambiar por ejemplo el stock.