

2023

Proyecto de BD eSports CSGO



Joaquín García Gutiérrez
IES ALIXAR
6-11-2023

Contenido

1. Introducción	3
<i>Versión 0.1</i>	3
<i>Versión 0.2</i>	3
<i>Version 0.3</i>	3
<i>Version 0.4</i>	3
2. Descripción	3
3. Modelo Entidad/Relación extendido y Modelo relacional.	5
1. MER.	5
2. Modelo relacional.	5
4. Paso a tablas en MySQL Workbench.	7
5. Importado a MySQL	7
6. Actualizaciones y modificaciones	7
<i>Modelo relacional:</i>	7
7. Carga de datos masiva	8
1. Tutorial de importación de csv para la carga masiva de datos.	8
8. Consultas SQL	13
<i>Objetivo:</i>	13
<i>SLQ:</i>	13
<i>Resultado:</i>	13
<i>Objetivo:</i>	14
<i>SLQ 2:</i>	14
<i>Resultado:</i>	14
<i>Objetivo:</i>	15
<i>SLQ 3:</i>	15
<i>Resultado:</i>	15
<i>Objetivo:</i>	15
<i>SLQ 4:</i>	16
<i>Resultado:</i>	16
<i>Objetivo:</i>	16
<i>SLQ 5:</i>	16
<i>Resultado:</i>	17
9. Vistas	17
<i>Vista de la consulta nº 1.</i>	17
SQL:	17
Resultado:	18
<i>Vista de la consulta nº 4.</i>	18

SQL:	18
Resultado:	19
10. Funciones y procedimientos	19
<i>Función 1:</i>	19
Objetivo:	19
SQL:	19
Resultado:	20
<i>Función 2:</i>	20
Objetivo:	20
SQL:	20
Resultado:	22
<i>Procedimiento 1:</i>	22
Objetivo:	22
SQL:	22
Resultado:	23
<i>Procedimiento 2:</i>	23
Objetivo:	23
SQL:	23
Resultado:	25
<i>Procedimiento 3:</i>	25
Objetivo:	25
SQL:	25
Resultado:	26
11. Triggers	27
<i>Trigger 1:</i>	27
Objetivo:	27
SQL:	27
Resultado:	28
<i>Trigger 2:</i>	28
Objetivo:	28
SQL:	28
Resultado:	29
12. Conclusión	29

1. 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.

Versión 0.3

Ahora que ya tenemos todas las tablas comenzaremos con el poblado masivo de estas, para hacer esto usaremos tanto [Google Datasheet](#) para poder modificar nuestras tablas a gusto e insertar los valores manualmente en caso de ser necesario, como [esta página](#) para generar masivamente los datos que puedan ser generados de forma automatizada.

Versión 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](#), junto con la imagen con el resultado.

2. Descripción

En el proyecto como ya hemos dicho en la [introducción](#) 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.

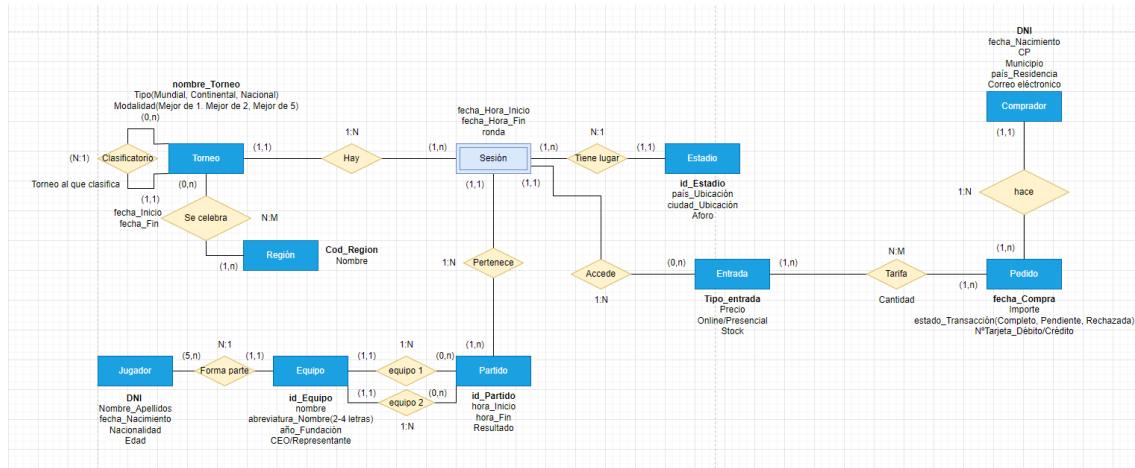
También necesitaremos guardar los pedidos, de cada pedido almacenaremos su fecha de compra, el importe, el estado de la transacción pudiendo ser completo pendiente o rechazado y el nº de tarjeta.

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.

3. Modelo Entidad/Relación extendido y Modelo relacional.

1. MER.



2. 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, N°Tarjeta_Débito/Crédito, DNI(Comprador))

PK: fecha_Compra(Pedido)

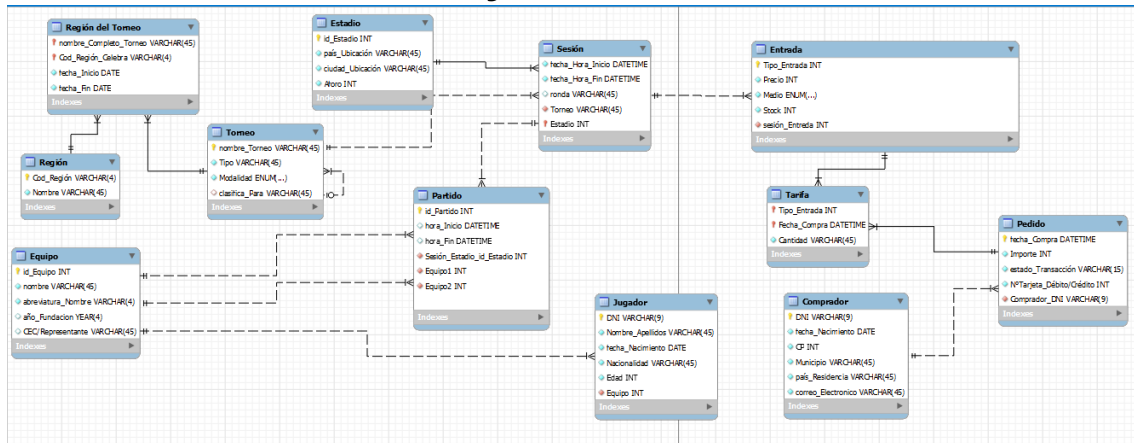
FK: DNI(Comprador)

Comprador (**DNI**, fecha_Nacimiento, CP, Municipio, país_Residencia, correo_Electronico)

PK: DNI(Comprador).



4. Paso a tablas en MySQL Workbench.



5. Importado a MySQL

Adjunto el archivo con el **Sql de la base de datos**.

Adjunto el archivo con el **Sql de la base de datos** y la carga de datos realizada.

6. 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, Sesión_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)

7. Carga de datos masiva

Para realizar la carga de datos masiva de nuestra base de datos usaremos la página web [onlinedatagenerator](#). 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.

Column Name	Field Type		
DNI	Personal Identification Number		
fecha_Nacimiento	Date (M/d/yyyy)		
CP	Integer		
Municipio	City		
país_Residencia	Country		
correo_Electronico	Email Address		

Add column Reset columns Save table definition* *User account required to save definition - create a free account [here](#)

2. Export data

Number of records to export

999

Export Type

CSV

Export

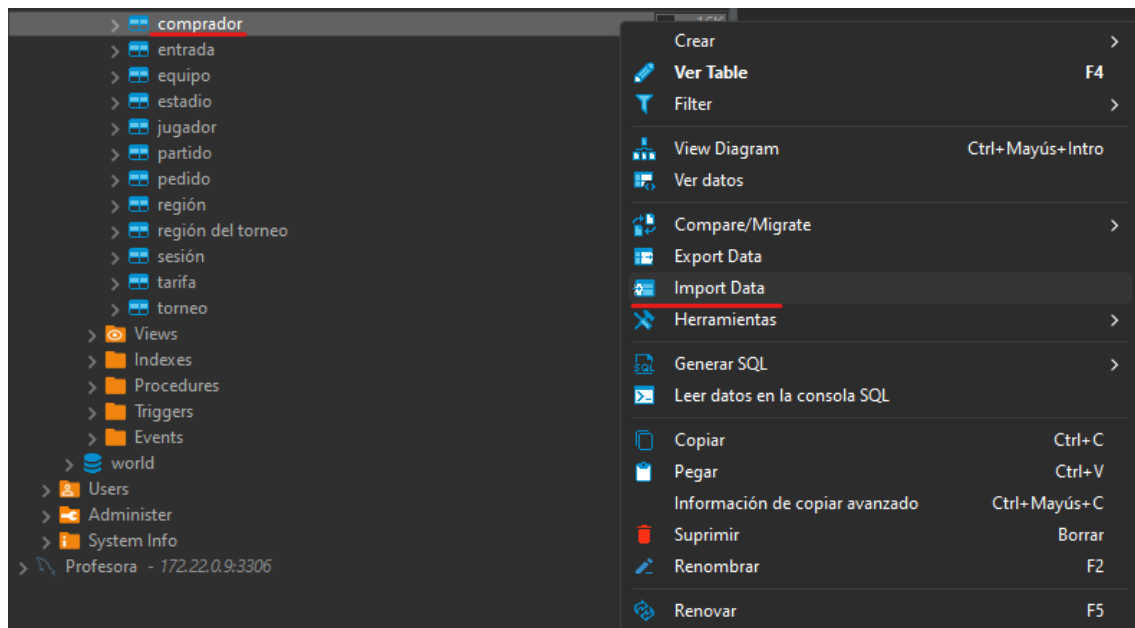
*(max 1000 rows with no account -> 100.000 with account)

Choose file type for the exported data

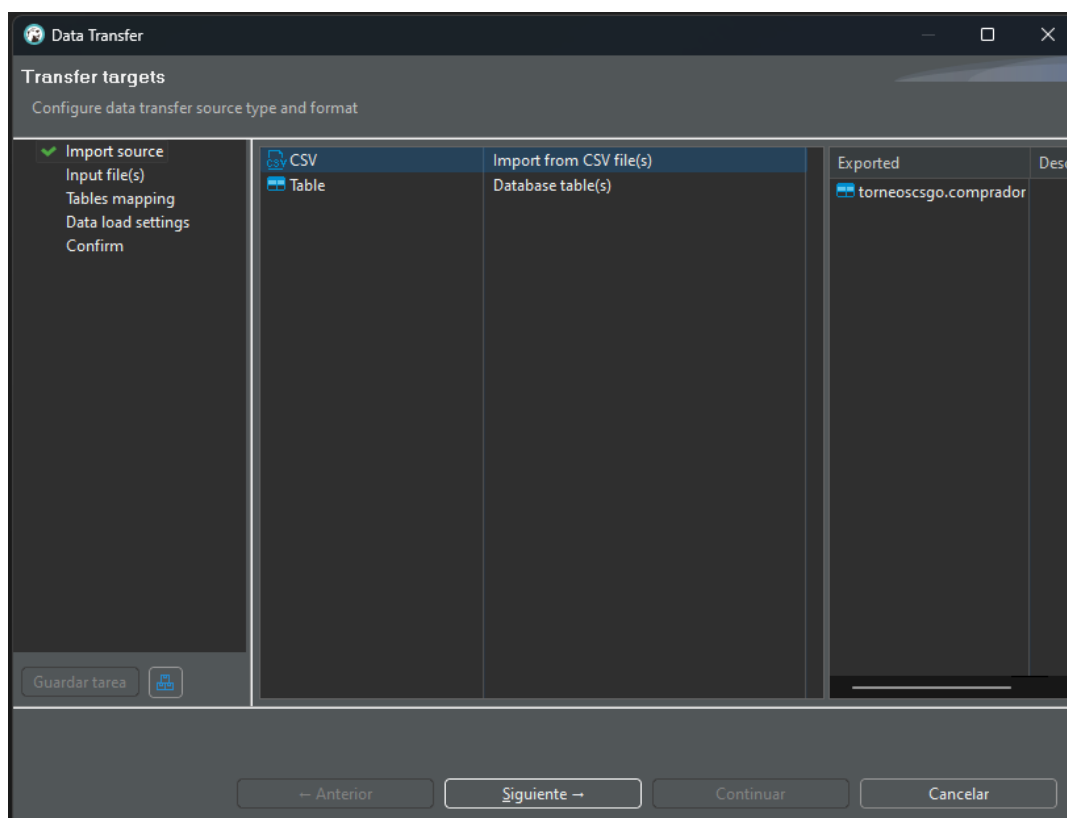
Además, como he dicho antes hemos podido limitar el integer, pues eso mismo haremos con la fecha de nacimiento la cual está establecida entre el 1/1/1950 y el 1/1/2005 esto lo haremos para evitar posibles compradores menores de edad y evidentemente que un bebe de 0 años pueda “comprar” una entrada. También 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.

1. Tutorial de importación de csv para la carga masiva de datos.

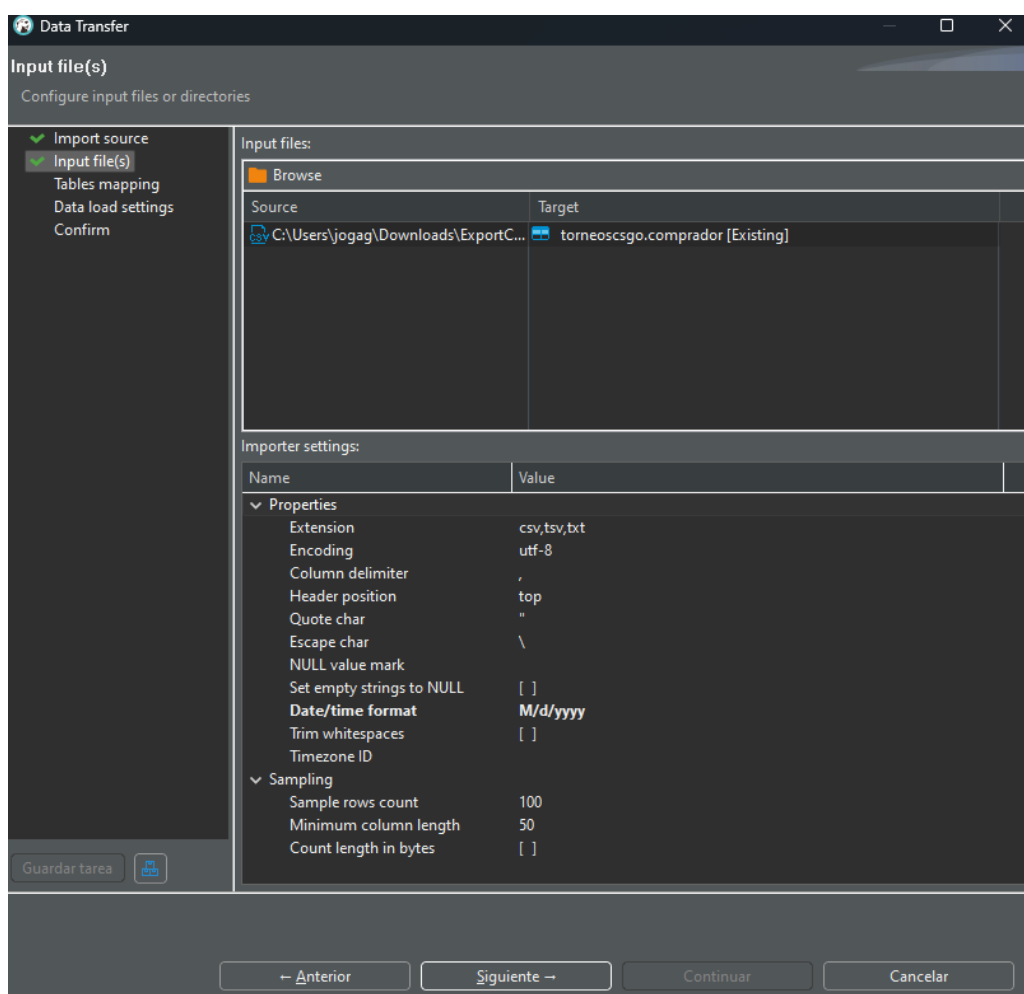
Una vez generado el csv con los datos, le daremos clic derecho en la tabla de comprador (en la que queremos insertar los datos), nos desplegara una serie de opciones y seleccionaremos “Import Data”.



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

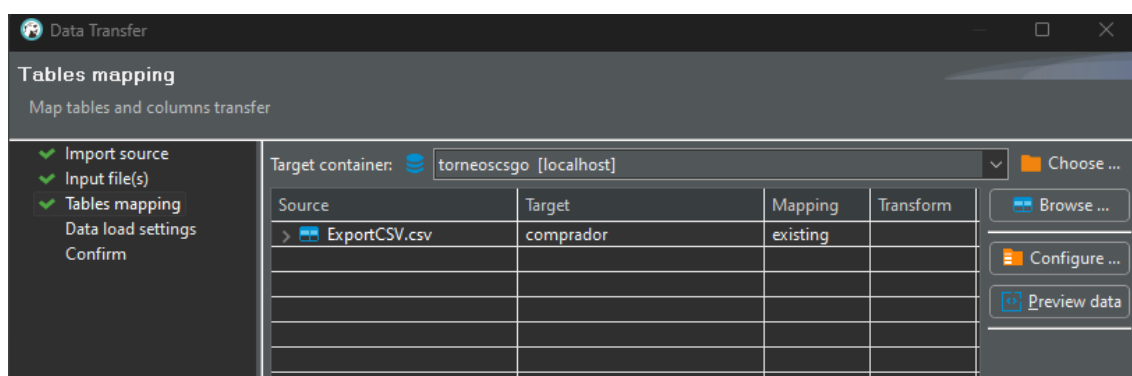


Al darle a siguiente nos abrirá el explorador de archivos donde seleccionaremos nuestro csv y nos saldrá lo siguiente:

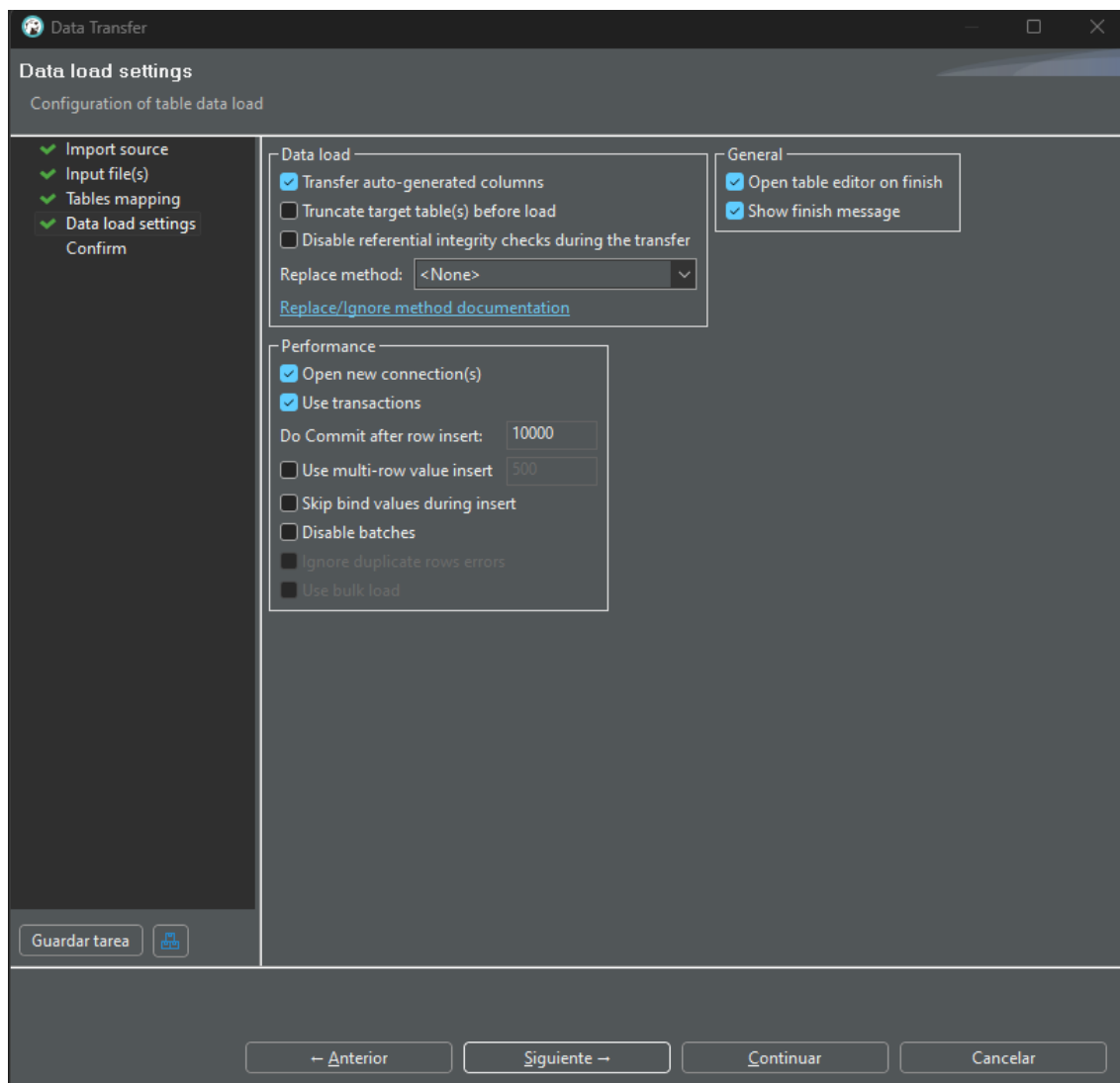


Aquí tendremos que configurar lo que nos pide, en mi caso solo cambio la fecha la cual modifico el formato a como se ha hecho en el csv, aunque en las próximas tablas que poblaremos necesitaremos activar el apartado de set empty strings to NULL ya que en esta tenemos configurado que no pueden ser valores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

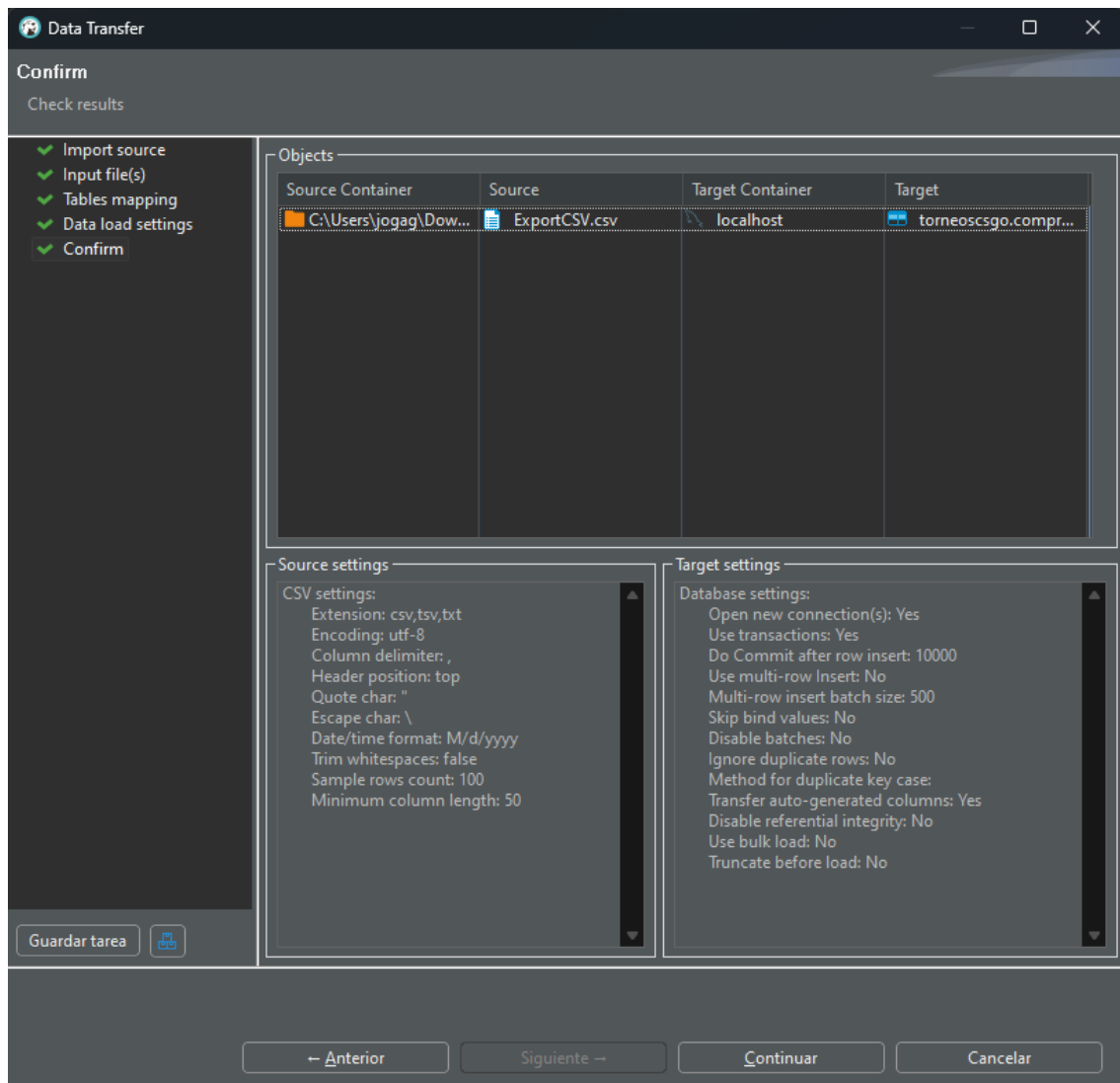
Una vez configurado seleccionaremos la base de datos en la que queramos añadirla y en este caso al ya haberla seleccionado anteriormente con el clic derecho no hará falta, pero en caso de no hacerlo así, en target pondremos la tabla y en Mapping al ya tener creada la tabla existing.



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:

comprador							
Propiedades Datos Diagrama ER							
comprador Enter a SQL expression to filter results (use Ctrl+Space)							
	DNI	fecha_Nacimiento	123 CP	ABC Municipio	ABC país_Residencia	ABC correo_Electronico	
1	000-02-5157	1992-05-22	99.648	San Jose	Myanmar	Harry_Driscoll5565@uagvw.ca	
2	000-34-2887	1992-07-21	98.859	Anaheim	Belgium	Maxwell_Allcott4064@qu9ml.design	
3	001-77-1418	1970-04-20	30.127	Miami	Vanuatu	Javier_Poole5024@d9un8.site	
4	001-83-5341	1976-12-02	11.048	Garland	Cameroon	Grace_Barrett1016@fhuux.zone	
5	002-24-4143	1960-02-12	97.798	Minneapolis	Uganda	Eduardo_Gordon3818@dbxli.business	
6	002-75-3800	2004-02-21	85.698	New Orleans	Malaysia	Christy_Booth5468@yvu30.net	
7	003-20-1735	1978-11-25	97.815	Long Beach	Chad	Michelle_Ramsey9184@p5emz.website	

Continuaremos poblando las tablas hasta haber poblado todas.

8. 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:

asc InfoEquipo ▼	123 Victorias_Nacionales ▼
ID: 4 Nombre: Team Liquid	3
ID: 8 Nombre: G2 Esports	3
ID: 9 Nombre: Made in Brazil	2
ID: 2 Nombre: Natus Vincere	1
ID: 6 Nombre: FaZe Clan	1
ID: 12 Nombre: OG Esports	1

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:

InfoEquipo	NumeroDeJugadores
Menores ID: 6 Nombre: FaZe Clan	1
Menores ID: 5 Nombre: Cloud9	1
Menores ID: 15 Nombre: Team Vitality	1
Menores ID: 3 Nombre: Fnatic	1
Mayores ID: 15 Nombre: Team Vitality	3
Mayores ID: 9 Nombre: Made in Brazil	2
Mayores ID: 4 Nombre: Team Liquid	1
Mayores ID: 14 Nombre: Ninjas in Pyjamas	1
Mayores ID: 8 Nombre: G2 Esports	1
Mayores ID: 1 Nombre: Astralis	1
Mayores ID: 6 Nombre: FaZe Clan	1
Mayores ID: 13 Nombre: BIG	1
Mayores ID: 11 Nombre: Renegades	1
Mayores ID: 10 Nombre: ENCE Esports	1

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.ResultadoEquip
o2) 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.ResultadoEquip
o2)>20
order by HorarioPartido desc;
```

Resultado:

HorarioPartido	NombreEquipo1	NombreEquipo2	TotalDeRondas
2023-01-15 21:30:00<>2023-01-15 22:34:59	Team Liquid	G2 Esports	21
2023-01-11 18:10:00<>2023-01-11 19:13:56	FaZe Clan	G2 Esports	25
2023-01-11 18:10:00<>2023-01-11 19:11:43	Natus Vincere	Team Liquid	21
2023-01-11 18:10:00<>2023-01-11 18:07:23	OG Esports	Made in Brazil	23
2023-01-07 17:05:00<>2023-01-07 17:43:36	Ninjas in Pyjamas	Made in Brazil	27
2023-01-05 17:50:00<>2023-01-05 18:31:02	Fnatic	FaZe Clan	29
2023-01-05 17:50:00<>2023-01-05 18:24:10	G2 Esports	Team Vitality	23
2023-01-03 19:40:00<>2023-01-03 20:33:45	Astralis	Natus Vincere	25
2023-01-03 19:40:00<>2023-01-03 20:21:33	Cloud9	Team Liquid	21

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:

asc Mes ▼	123 Dinero_Generado ▼	123 Pedidos_Realizados ▼
3-March	51.375	94
11-November	49.361	89
5-May	47.924	93
9-September	47.516	86
7-July	45.916	91
4-April	42.080	79
12-December	41.791	83
8-August	41.453	82
10-October	40.891	79
2-February	37.257	71
1-January	36.741	79
6-June	33.315	73

RIKE®
L OFFENSIVE

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.país 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.país Residencia
ORDER BY Cantidad Ventas DESC;
```

Resultado:

País	Cantidad_Ventas
Russia	6
Botswana	6
Bolivia	6
Cambodia	6
Equatorial Guinea	6
Bulgaria	6
Kuwait	6
Malaysia	6
Solomon Islands	6
Seychelles	6
Togo	4
Cameroon	4
Czech Republic	4
Haiti	4
Saint Lucia	4
Angola	4
Mozambique	4

9. Vistas

Vista de la consulta nº 1.

SQL:

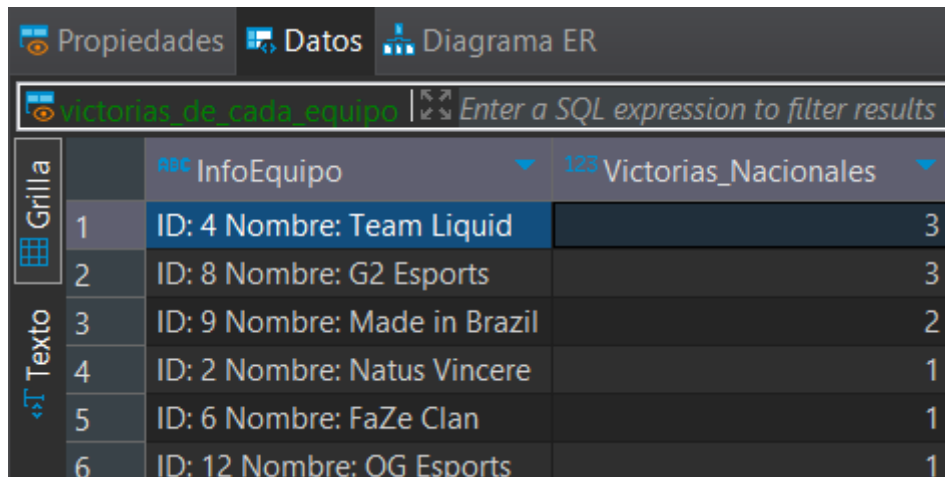
```
CREATE VIEW Victorias_de_cada_equipo AS
select concat("ID: ", e.id_Equipo, "
Nombre: ", e.nombre) AS 'InfoEquipo',
COUNT(p.id_Partido) AS
'Victorias_Nacionales' from equipo e
inner join partido p ON e.id_Equipo =
p.Equipo1 OR e.id_Equipo = p.Equipo2
inner join sesión s ON
p.Sesion_Estadio_id_Estadio = 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:



The screenshot shows a database application with three tabs: 'Propiedades', 'Datos', and 'Diagrama ER'. The 'Datos' tab is active, displaying a table named 'victorias_de_cada_equipo'. The table has two columns: 'InfoEquipo' and 'Victorias_Nacionales'. The 'InfoEquipo' column contains team IDs and names, and the 'Victorias_Nacionales' column contains the number of national victories. The table is sorted by 'Victorias_Nacionales' in descending order.

	InfoEquipo	Victorias_Nacionales
1	ID: 4 Nombre: Team Liquid	3
2	ID: 8 Nombre: G2 Esports	3
3	ID: 9 Nombre: Made in Brazil	2
4	ID: 2 Nombre: Natus Vincere	1
5	ID: 6 Nombre: FaZe Clan	1
6	ID: 12 Nombre: OG Esports	1

Vista de la consulta nº 4.

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:

Propiedades Datos Diagrama ER				
Ingresos y pedidos totales por mes Enter a SQL expression to filter results (u				
Grilla Texto		ABC Mes ▼	123 Dinero_Generado ▼	123 Pedidos_Realizados ▼
	1	3-March	51.375	94
	2	11-November	49.361	89
	3	5-May	47.924	93
	4	9-September	47.516	86
	5	7-July	45.916	91
	6	4-April	42.080	79
	7	12-December	41.791	83
	8	8-August	41.453	82
	9	10-October	40.891	79
	10	2-February	37.257	71
	11	1-January	36.741	79
	12	6-June	33.315	73

10. 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');
```

123 calcularIngresosTorneo('eSL Liga')	
	39.610

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');
```

123 calcular_porcentaje_victorias_equipo_en_torneo('6','eSL Liga')

50

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');

ABC Mensaje

Se ha creado un nuevo equipo con ID 17, nombre Prueba nuevo equipo y abreviatura PNE

CALL

gestionar_informacion_equipo('Prueba
nuevo equipo', 'PNN');

ABC Mensaje

Se ha actualizado la abreviatura del equipo Prueba nuevo equipo a 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');
```

ASC Total_Ingresos	ASC Total_Pedidos
Los ingresos totales del torneo eSL Liga son: 39610.00€.	El número total de pedidos realizados en el torneo eSL Liga es: 691

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');

```

Informacion_Jugadores		
106-73-3066	242-48-0232	403-22-0417
577-87-2441	831-48-0521	
106-73-3066		
242-48-0232		
403-22-0417		
577-87-2441		
831-48-0521		

11. 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:

Grilla	fecha_compra	accion	fecha_modificacion
1	2023-04-30 22:33:00	Estado cambiado	2024-04-07 09:51:37

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:

2023-04-30 22:33:00	47	Procesando	3254-7621-2161-0151	662-31-1538	Estandar	3	30
Estandar	24,99	presencial	1.997	30			

12. 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 de 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.