

# Proyecyo Final

Raya Pérez José Luis

Muñoz Sandoval Alan Sebastián

Cruz Martínez Giovanni

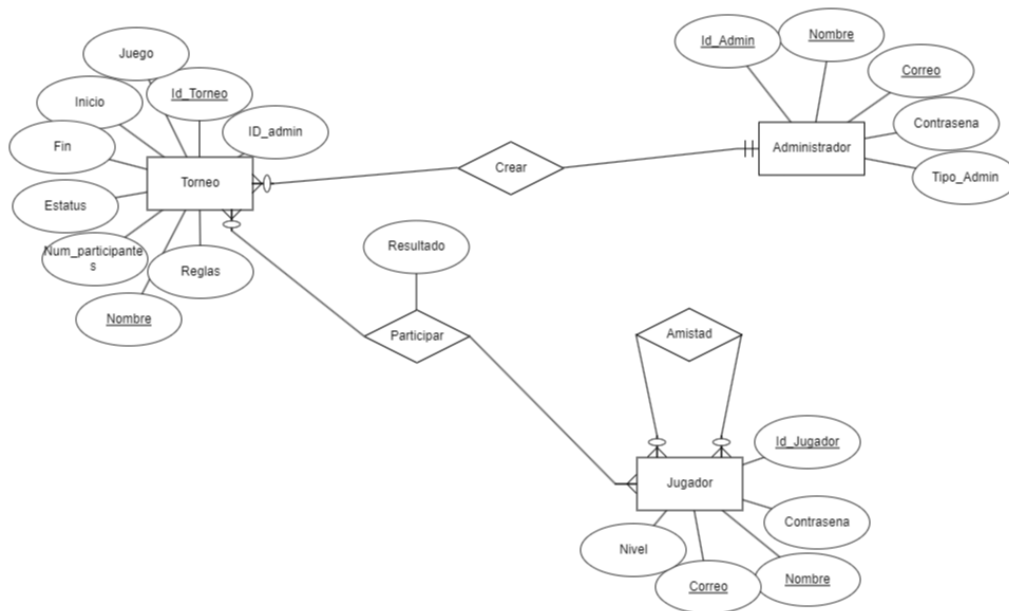
Diciembre 2023

## Base de Datos para administración de torneos MASTER GAME

### 1. Requerimientos

1. Gestión de torneos:  
Manejo de los torneos creados por los administradores. Se tienen detalles como Id\_Torneo, ID\_Admin, Nombre, Juego, Reglas, Numero de participantes, Estatus, Inicio del torneo y Fin del torneo.
  2. Gestión de los administradores:  
Se maneja información de los administradores. ID\_Admin, Nombre, Correo, Contraseña, Tipo\_Admin. Hay dos tipos de administradores, los activos y los inactivos.
  3. Registro de los jugadores:  
Se tendrá información de los jugadores como lo es Id\_Jugador, Nombre, Contraseña y Nivel.
  4. Participar:  
Los jugadores se pueden registrar para los torneos y jugar obteniendo un resultado. Se registrara el torneo, el jugador y el resultado.
  5. Amistad:  
Los jugadores podran tener amigos, para esto se almacenara el id de jugador y el id del amigo.
- Almacenamiento eficiente: La base de datos debe poder almacenar cantidades considerables de datos, de una manera eficiente y garantizar la integridad de la información.
  - Gestión de relaciones: Debe permitir establecer, gestionar las relaciones que existen entre las entidades. Así como la asociación que hay entre los torneos, el administrador que lo creo, el jugador que participa, además de las amistades entre los jugadores.
  - Consultas eficientes: Se debe poder realizar consultas con una alta eficiencia, así como búsqueda de torneos, relaciones de amistad, administradores o resultados.
  - Seguridad: Debe contar con mecanismos que fortalezcan la seguridad de la base.
  - Escalabilidad: La base de datos debe soportar grandes cantidades de información y así ser útil ante el crecimiento de los usuarios posibles a futuro.
  - Integridad de los datos. Debe garantizar la integridad de los datos, evitando tener inconsistencias. Así como la precisión y confiabilidad de los datos.
- Este listado de requerimientos son pieza clave para el correcto funcionamiento de la base de datos, para poder gestionar la participación de los jugadores en los torneos, y poder establecer una correcta administración de los torneos, la participación, y las amistades dentro de la plataforma.

## 2. Modelo Conceptual



### Administrador:

- ID\_Admin: Identificador único del administrador (Llave primaria).
- Nombre: Nombre del administrador.
- Correo: Correo único del administrador.
- Contraseña: Contraseña del administrador.
- Tipo\_Admin: Tipo de administrador (1 para superadmin, 2 para admin regular).

### Torneo:

- ID\_Torneo: Identificador único del torneo (Llave primaria).
- ID\_admin: Identificador del administrador que organiza el torneo.
- Nombre: Nombre único del torneo.
- Juego: Juego asociado al torneo. : Reglas del torneo.
- Num\_participantes: Número de participantes permitidos en el torneo.
- Estatus: Estado del torneo (1 para planificado, 2 para en curso, 3 para completado).
- Inicio: Fecha de inicio del torneo.
- Fin: Fecha de finalización del torneo (debe ser antes de la fecha de inicio).

### Jugador:

- Nombre: Nombre del jugador.
- Id\_Jugador: Identificador único del jugador (Llave primaria).
- Correo: Correo único del jugador.

- Contraseña: Contraseña del jugador.
- Nivel: Nivel de habilidad del jugador (en una escala del 1 al 100).

### Participar:

- Resultado: Resultado de la participación del jugador en el torneo.
- Id\_Torneo: Identificador del torneo en el que el jugador participa (Parte de la clave primaria).
- Id\_Jugador: Identificador del jugador que participa (Parte de la clave primaria).

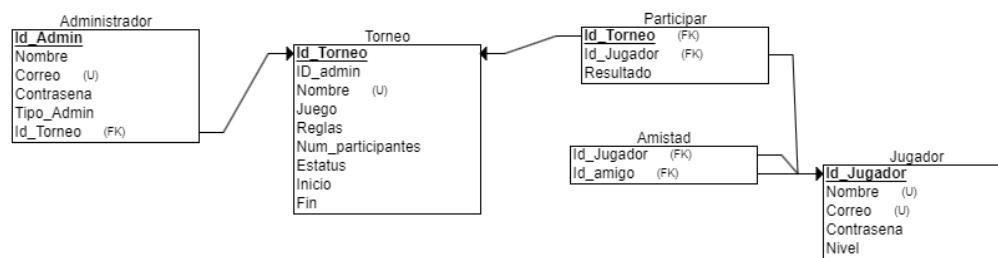
### Amistad:

- Id\_Jugador: Identificador del jugador (Parte de la clave primaria).
- Id\_amigo: Identificador del amigo (Parte de la clave primaria).

## Relaciones

- **Participa (Jugador - Torneo):** Un jugador participa en un torneo .
- **Amistad (Jugador - Jugador):** Un jugador puede agregar a otro jugador a su lista de amigos.

## 3. Modelo Relacional



## 4. Evidencia del funcionamiento de al menos 4 restricciones de integridad referencial.

Evidencia 1.

1. Tablas involucradas en la restricción:  
ADMINISTRADOR - TORNEO
2. FK de la tabla que referencia y PK de la tabla referenciada.

FK ID\_admin. (TORNEO)  
PK ID\_admin. (ADMINISTRADOR)

3. Justificación del trigger de integridad referencial elegido.

Se escogio ON DELETE SET DEFAULT debido a que si un administrador es eliminado, el torneo se al súper administrador que tiene asignado el indice 1.

4. Instrucción UPDATE o DELETE que permita evidenciar que la restricción está funcionando.

```
DELETE FROM ADMINISTRADOR
WHERE id_admin = 2;
```

5. Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.

| Data Output Messages Notifications |                           |                     |                                   |                                  |                                      |                              |                                   |                |             |
|------------------------------------|---------------------------|---------------------|-----------------------------------|----------------------------------|--------------------------------------|------------------------------|-----------------------------------|----------------|-------------|
|                                    | id_torneo<br>[PK] integer | id_admin<br>integer | nombre<br>character varying (255) | juego<br>character varying (255) | reglas<br>character varying (255)    | num_participantes<br>integer | estatus<br>character varying (15) | inicio<br>date | fin<br>date |
| 1                                  | 31                        | 2                   | Evelyn Prentice                   | Drama Mystery Romance            | engage B2B technologies              | 27                           | Finalizado                        | 2023-12-29     | 2024-03-10  |
| 2                                  | 61                        | 2                   | Shark in Venice                   | Action Horror Thriller           | reinvent best-of-breed architectures | 33                           | Finalizado                        | 2023-12-30     | 2024-03-28  |

| Query Query History |                           |  |  |  |  |  |  |  |  |
|---------------------|---------------------------|--|--|--|--|--|--|--|--|
| 1                   | DELETE FROM ADMINISTRADOR |  |  |  |  |  |  |  |  |
| 2                   | WHERE id_admin = 2;       |  |  |  |  |  |  |  |  |
| 3                   |                           |  |  |  |  |  |  |  |  |
| 4                   | SELECT * FROM Torneo      |  |  |  |  |  |  |  |  |
| 5                   | Where id_admin =1 ;       |  |  |  |  |  |  |  |  |

| Data Output Messages Notifications |                           |                     |                                   |                                  |                                      |                              |                                   |                |             |
|------------------------------------|---------------------------|---------------------|-----------------------------------|----------------------------------|--------------------------------------|------------------------------|-----------------------------------|----------------|-------------|
|                                    | id_torneo<br>[PK] integer | id_admin<br>integer | nombre<br>character varying (255) | juego<br>character varying (255) | reglas<br>character varying (255)    | num_participantes<br>integer | estatus<br>character varying (15) | inicio<br>date | fin<br>date |
| 1                                  | 31                        | 1                   | Evelyn Prentice                   | Drama Mystery Romance            | engage B2B technologies              | 27                           | Finalizado                        | 2023-12-29     | 2024-03-10  |
| 2                                  | 61                        | 1                   | Shark in Venice                   | Action Horror Thriller           | reinvent best-of-breed architectures | 33                           | Finalizado                        | 2023-12-30     | 2024-03-28  |

## Evidencia 2.

1. Tablas involucradas en la restricción:  
TORNEO - PARTICIPAR
2. FK de la tabla que referencia y PK de la tabla referenciada.

FK ID\_Torneo. (JUGADOR)  
PK ID\_Torneo. (PARTICIPAR)

3. Justificación del trigger de integridad referencial elegido.

Se escogió CASCADE ya que si eliminamos o modificamos algún torneo se deben eliminar los registros de participación de los jugadores registrados o eliminar los registros.

4. Instrucción UPDATE o DELETE que permita evidenciar que la restricción está funcionando.  
DELETE FROM Torneo  
WHERE id\_torneo = 13;
5. Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.

|                                    |                           |                            |                                     |
|------------------------------------|---------------------------|----------------------------|-------------------------------------|
| Query Query History                |                           |                            |                                     |
| 1 SELECT * FROM Participar;        |                           |                            |                                     |
| Data Output Messages Notifications |                           |                            |                                     |
|                                    | id_torneo<br>[PK] integer | id_jugador<br>[PK] integer | resultado<br>character varying (10) |
| 1                                  | 13                        | 96                         | Perdedor                            |
| 2                                  | 25                        | 61                         | Ganador                             |
| 3                                  | 99                        | 74                         | En curso                            |
| 4                                  | 90                        | 4                          | En curso                            |
| 5                                  | 22                        | 65                         | Perdedor                            |
| 6                                  | 5                         | 25                         | Ganador                             |
| 7                                  | 89                        | 35                         | En curso                            |

|                                    |                           |                            |                                     |
|------------------------------------|---------------------------|----------------------------|-------------------------------------|
| Query Query History                |                           |                            |                                     |
| 1 DELETE FROM Torneo               |                           |                            |                                     |
| 2 WHERE id_torneo = 13;            |                           |                            |                                     |
| 3                                  |                           |                            |                                     |
| 4 SELECT * FROM Participar;        |                           |                            |                                     |
| Data Output Messages Notifications |                           |                            |                                     |
|                                    | id_torneo<br>[PK] integer | id_jugador<br>[PK] integer | resultado<br>character varying (10) |
| 1                                  | 25                        | 61                         | Ganador                             |
| 2                                  | 99                        | 74                         | En curso                            |
| 3                                  | 90                        | 4                          | En curso                            |
| 4                                  | 22                        | 65                         | Perdedor                            |
| 5                                  | 5                         | 25                         | Ganador                             |
| 6                                  | 89                        | 35                         | En curso                            |
| 7                                  | 36                        | 11                         | Perdedor                            |
| 8                                  | 90                        | 10                         | Ganador                             |
| 9                                  | 96                        | 32                         | En curso                            |
| 10                                 | 21                        | 22                         | Perdedor                            |

## Evidencia 3.

1. Tablas involucradas en la restricción:  
JUGADOR - PARTICIPAR
2. FK de la tabla que referencia y PK de la tabla referenciada.  
FK ID\_jugador. (JUGADOR)  
PK ID\_jugador. (PARTICIPAR)
3. Justificación del trigger de integridad referencial elegido.  
Se escogió CASCADE ya que si eliminamos a un jugador se debe eliminar de todos los torneos en los que este participando.
4. Instrucción UPDATE o DELETE que permita evidenciar que la restricción está funcionando.  
DELETE FROM Jugador  
WHERE id\_jugador = 61;

5. Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.

The screenshot shows a database management tool interface. At the top, a SQL query is entered in the editor:

```
3
4 SELECT * FROM Participar;
```

Below the editor, the 'Data Output' tab is selected, displaying the results of the query. The results are shown in a table with the following columns: `id_torneo` (PK integer), `id_jugador` (PK integer), and `resultado` (character varying (10)).

|    | id_torneo<br>[PK] integer | id_jugador<br>[PK] integer | resultado<br>character varying (10) |
|----|---------------------------|----------------------------|-------------------------------------|
| 1  | 25                        | 61                         | Ganador                             |
| 2  | 99                        | 74                         | En curso                            |
| 3  | 90                        | 4                          | En curso                            |
| 4  | 22                        | 65                         | Perdedor                            |
| 5  | 5                         | 25                         | Ganador                             |
| 6  | 89                        | 35                         | En curso                            |
| 7  | 36                        | 11                         | Perdedor                            |
| 8  | 90                        | 10                         | Ganador                             |
| 9  | 96                        | 32                         | En curso                            |
| 10 | 91                        | 22                         | Perdedor                            |

Below the table, the 'Query History' tab is selected, showing the executed queries:

```
1 DELETE FROM Jugador
2 WHERE id_jugador = 61;
3
4 SELECT * FROM Participar;
```

The screenshot also shows a second instance of the 'Data Output' tab, which displays the results of the query after the deletion of the row with `id_jugador = 61`. The results are shown in a table with the following columns: `id_torneo` (PK integer), `id_jugador` (PK integer), and `resultado` (character varying (10)).

|   | id_torneo<br>[PK] integer | id_jugador<br>[PK] integer | resultado<br>character varying (10) |
|---|---------------------------|----------------------------|-------------------------------------|
| 1 | 99                        | 74                         | En curso                            |
| 2 | 90                        | 4                          | En curso                            |
| 3 | 22                        | 65                         | Perdedor                            |
| 4 | 5                         | 25                         | Ganador                             |
| 5 | 89                        | 35                         | En curso                            |
| 6 | 36                        | 11                         | Perdedor                            |
| 7 | 90                        | 10                         | Ganador                             |
| 8 | 96                        | 32                         | En curso                            |
| 9 | 91                        | 22                         | Perdedor                            |

#### Evidencia 4.

1. Tablas involucradas en la restricción:  
JUGADOR - AMISTAD
2. FK de la tabla que referencia y PK de la tabla referenciada.  
  
FK ID\_jugador. (JUGADOR)  
PK ID\_jugador. (AMISTAD)
3. Justificación del trigger de integridad referencial elegido.  
Se escogió CASCADE ya que si eliminamos a un jugador o modificamos su clave, entonces esto debe afectar a todas las filas donde este involucrado en la tabla AMISTAD.
4. Instrucción UPDATE o DELETE que permita evidenciar que la restricción está funcionando.  
DELETE FROM JUGADOR

WHERE id\_jugador =55;

5. Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.

QueryQuery History

1SELECT \* FROM AMISTAD;

Data OutputMessagesNotifications

|    | id_jugador<br>integer | id_amigo<br>integer |
|----|-----------------------|---------------------|
| 1  | 55                    | 64                  |
| 2  | 79                    | 28                  |
| 3  | 74                    | 95                  |
| 4  | 13                    | 25                  |
| 5  | 60                    | 55                  |
| 6  | 28                    | 32                  |
| 7  | 46                    | 16                  |
| 8  | 22                    | 40                  |
| 9  | 1                     | 8                   |
| 10 | 49                    | 47                  |
| 11 | 62                    | 77                  |
| 12 | 13                    | 30                  |

QueryQuery History

1DELETE FROM JUGADOR  
2WHERE id\_jugador =55;  
3  
4SELECT \* FROM AMISTAD;

Data OutputMessagesNotifications

|    | id_jugador<br>integer | id_amigo<br>integer |
|----|-----------------------|---------------------|
| 1  | 79                    | 28                  |
| 2  | 74                    | 95                  |
| 3  | 13                    | 25                  |
| 4  | 28                    | 32                  |
| 5  | 46                    | 16                  |
| 6  | 22                    | 40                  |
| 7  | 1                     | 8                   |
| 8  | 49                    | 47                  |
| 9  | 62                    | 77                  |
| 10 | 13                    | 30                  |
| 11 | 97                    | 66                  |
| 12 | 63                    | 89                  |

## 5. Evidencia del funcionamiento de al menos 3 restricciones check para “atributos” de varias tablas.

### Evidencia 1

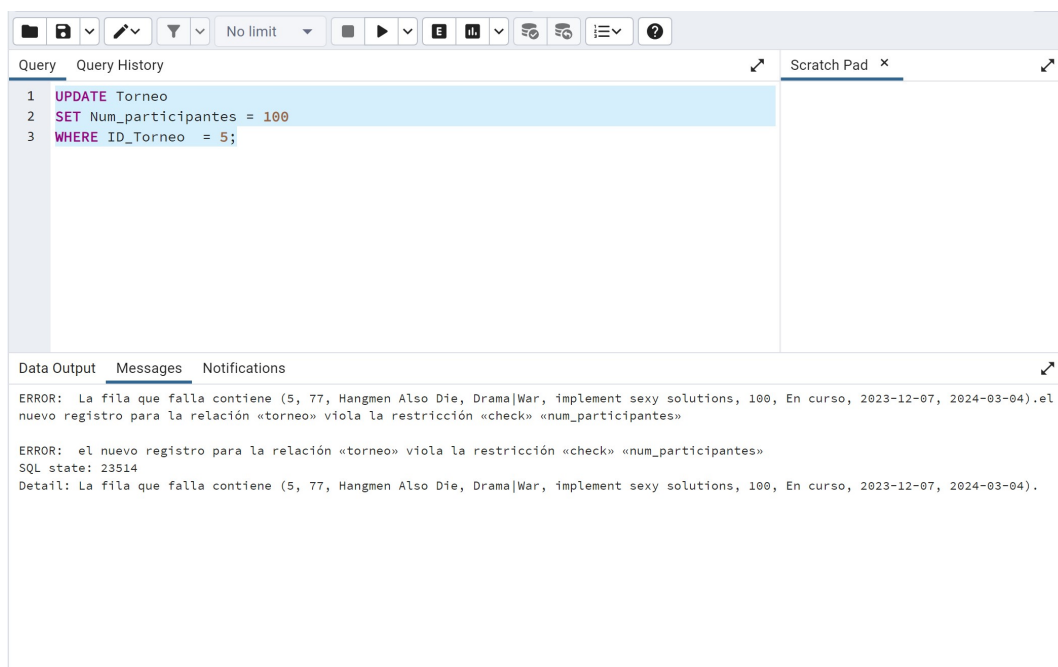
1. Tabla elegida.  
TORNEO
2. Atributo elegido  
Num\_participantes
3. Breve descripción de la restricción Se debe cumplir con un mínimo de 10 participantes por torneo y un máximo de 50.
4. Instrucción para la creación de la restricción.  

```
ALTER TABLE Torneo  
ADD CONSTRAINT Num_participantes  
CHECK (Num_participantes BETWEEN 10 AND 50);
```
5. Instrucción que permita evidenciar que la restricción esta funcionando.

Listing 1: Instrucción que evidencia funcionamiento

```
UPDATE Torneo  
SET Num_participantes = 100  
WHERE ID_Torneo = 5;
```

6. Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.



### Evidencia 2

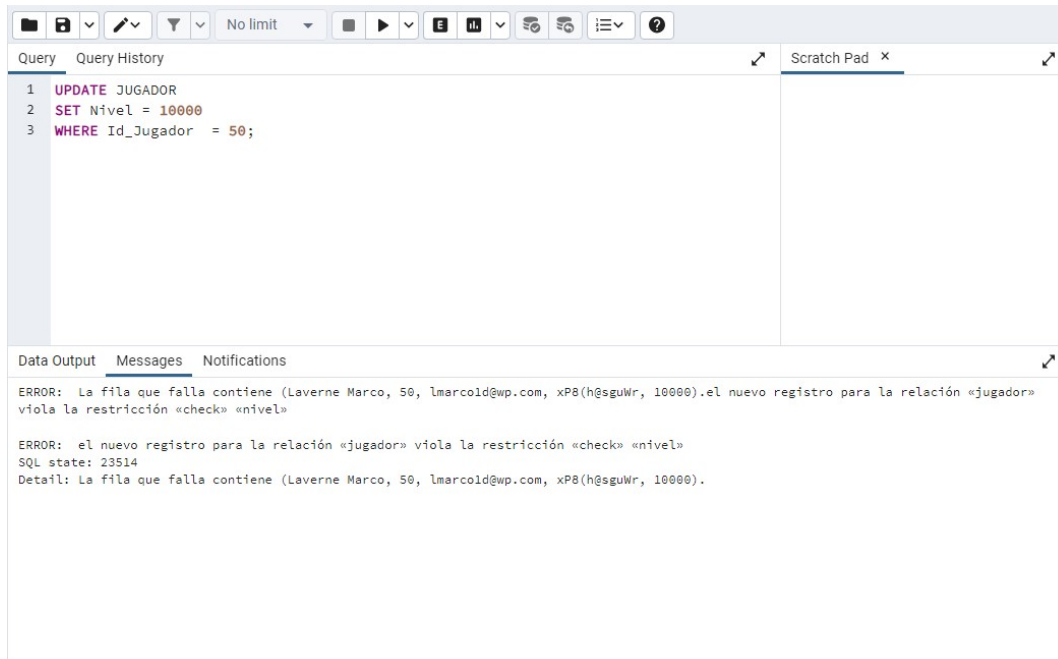
1. Tabla elegida.  
JUGADOR
2. Atributo elegido  
Nivel



- Breve descripción de la restricción  
El nivel de un jugador debe ir entre 1 y 100.
- Instrucción para la creación de la restricción.

```
ALTER TABLE Jugador ADD CONSTRAINT Nivel  
CHECK (Nivel BETWEEN 1 AND 100);
```

- Instrucción que permita evidenciar que la restricción esta funcionando.
- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.



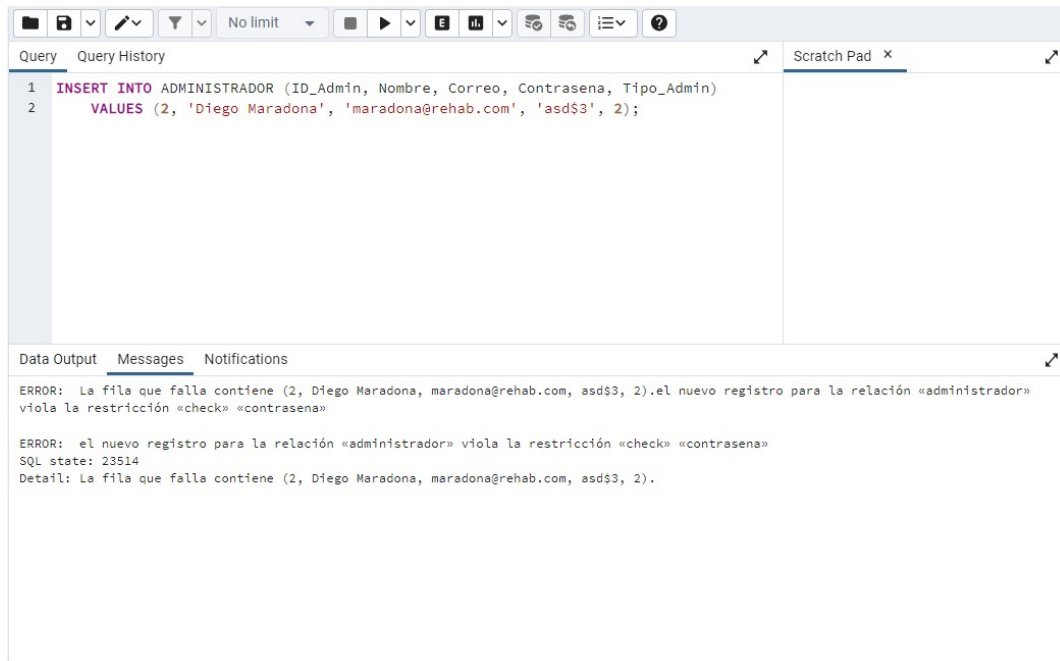
### Evidencia 3

- Tabla elegida.  
Administrador
- Atributo elegido  
Contraseña
- Breve descripción de la restricción  
La contraseña debe tener al menos 8 caracteres.
- Instrucción para la creación de la restricción.  

```
ALTER TABLE Administrador  
ADD CONSTRAINT Contraseña  
CHECK (LENGTH(Contraseña) >= 8);
```
- Instrucción que permita evidenciar que la restricción esta funcionando.

```
INSERT INTO ADMINISTRADOR (ID_Admin, Nombre, Correo, Contraseña, Tipo_Admin)  
VALUES (2, 'Diego-Maradona', 'maradona@rehab.com', 'asd$3 ', 2);
```

6. Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.



## 6. Evidencia de la creación de al menos tres dominios personalizados. Se deben utilizar restricciones check en la creación de los tres dominios.

Evidencia 1.

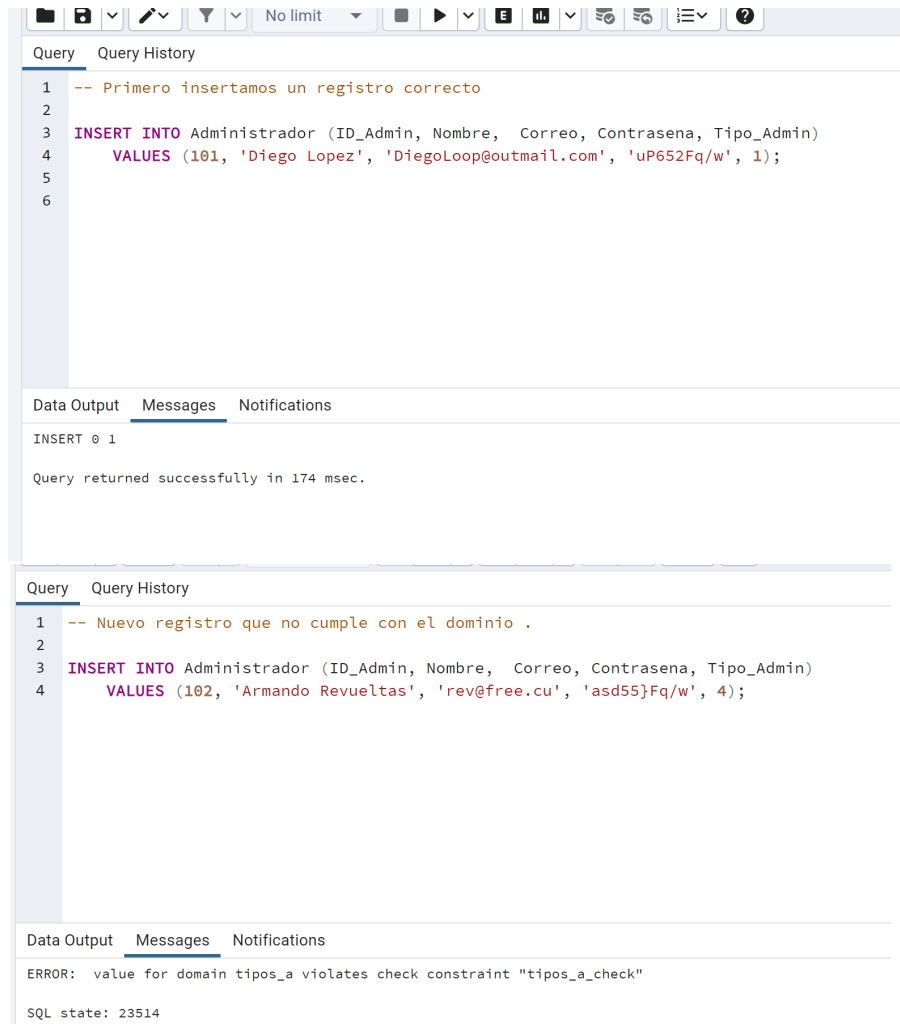
1. Tabla elegida.  
ADMINISTRADOR
2. Atributo elegido  
Tipo\_admin
3. Breve descripción del dominio y de la restricción check propuesta.

Como solo pueden existir dos tipos de administrador, cuando el tipo es 1 es un administrador activo y cuando es 2 es un administrador inactivo.

4. Instrucción para la creación del dominio personalizado.

```
CREATE DOMAIN Tipos AS INT  
CHECK(Tipos BETWEEN 1 AND 2);
```

5. Captura de pantalla de la estructura de la tabla donde se muestre el dominio personalizado en uso.



The image displays two screenshots of a SQL query editor interface, likely DBeaver, showing the execution of SQL queries and the resulting messages.

**Top Screenshot:**

- Query:**

```
1 -- Primero insertamos un registro correcto
2
3 INSERT INTO Administrador (ID_Admin, Nombre, Correo, Contraseña, Tipo_Admin)
4 VALUES (101, 'Diego Lopez', 'DiegoLoop@outmail.com', 'uP652Fq/w', 1);
5
6
```
- Messages:**

```
INSERT 0 1
Query returned successfully in 174 msec.
```

**Bottom Screenshot:**

- Query:**

```
1 -- Nuevo registro que no cumple con el dominio .
2
3 INSERT INTO Administrador (ID_Admin, Nombre, Correo, Contraseña, Tipo_Admin)
4 VALUES (102, 'Armando Revueltas', 'rev@free.cu', 'asd55jFq/w', 4);
```
- Messages:**

```
ERROR: value for domain tipos_a violates check constraint "tipos_a_check"
SQL state: 23514
```

## Evidencia 2.

1. Tabla elegida.  
TORNEO
2. Atributo elegido  
Estatus
3. Breve descripción del dominio y de la restricción check propuesta. Solo existen tres estatus validos para un torneo.
  - Disponible
  - En Curso
  - Finalizado

Por lo tanto no es aceptable que exista otro estatus.

4. Instrucción para la creación del dominio personalizado.

```
CREATE DOMAIN Tipos_estatus AS VARCHAR(15)  
CHECK(Tipos_estatus IN ('En curso', 'Finalizado','Disponible'));
```

5. Captura de pantalla de la estructura de la tabla donde se muestre el dominio personalizado en uso.

The screenshot displays the SQL Developer interface with two query windows. The first window shows a successful INSERT query for the TORNEO table. The second window shows an INSERT query that failed due to a domain constraint violation.

```
Query    Query History  
1  -- Primero insertamos un registro correcto  
2  
3  INSERT INTO TORNEO (Id_Torneo, ID_Admin, Nombre, Juego, Reglas, Num_participantes, Estatus, Inicio, Fin)  
4    VALUES (101, 60, 'Fifa WORLD', 'FIFA 2022', '1 vs 1', 20, 'Disponible', '2023-10-20 10:47:27', '2024-01-01 22:47:09');  
5  
6  
  
Data Output  Messages  Notifications  
INSERT 0 1  
Query returned successfully in 162 msec.  
  
Query    Query History  
1  -- Nuevo registro que no cumple con el dominio .  
2  
3  INSERT INTO TORNEO (Id_Torneo, ID_Admin, Nombre, Juego, Reglas, Num_participantes, Estatus, Inicio, Fin)  
4    VALUES (102, 40, 'PESS WORLD', 'PESS 2022', '1 vs 1', 30, 'CANCELADO', '2023-10-20 10:47:27', '2024-01-01 22:47:09');  
  
Data Output  Messages  Notifications  
ERROR:  value for domain tipos_estatus violates check constraint "tipos_estatus_check"  
SQL state: 23514
```

## Evidencia 3.

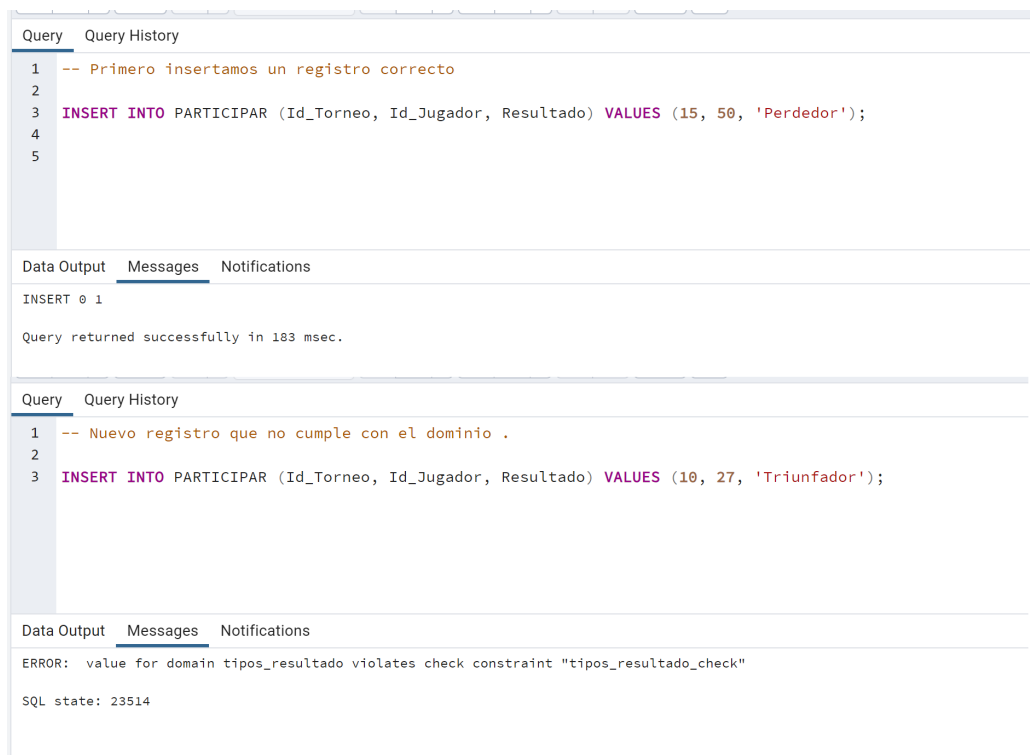
1. Tabla elegida.  
PARTICIPAR
2. Atributo elegido.  
Resultado
3. Breve descripción del dominio y de la restricción check propuesta.

En esta caso solo se puede tener 3 tipos de estatus de un jugador en un torneo, si el torneo aun no finaliza entonces el jugador aun no tiene un resultado por lo que el estatus sera .<sup>En</sup> curso”, una vez que finalice el mismo tendrá un estatus ya sea que haya ganado o perdido. Por lo tanto es importante que solo existan estos estatus en el resultado.

4. Instrucción para la creación del dominio personalizado.

```
CREATE DOMAIN Tipos_resultado AS VARCHAR(10)
CHECK(Tipos_resultado IN ('Ganador', 'Perdedor', 'En curso'));
```

5. Captura de pantalla de la estructura de la tabla donde se muestre el dominio personalizado en uso.



```
Query    Query History
1  -- Primero insertamos un registro correcto
2
3  INSERT INTO PARTICIPAR (Id_Torneo, Id_Jugador, Resultado) VALUES (15, 50, 'Perdedor');
4
5

Data Output    Messages    Notifications
INSERT 0 1
Query returned successfully in 183 msec.

Query    Query History
1  -- Nuevo registro que no cumple con el dominio .
2
3  INSERT INTO PARTICIPAR (Id_Torneo, Id_Jugador, Resultado) VALUES (10, 27, 'Triunfador');

Data Output    Messages    Notifications
ERROR: value for domain tipos_resultado violates check constraint "tipos_resultado_check"
SQL state: 23514
```

## 7. Evidencia del funcionamiento de al menos 2 restricciones para “tuplas” en diferentes tablas (Unidad 8 Integridad, tema “Specifying Constraints on Tuples Using CHECK”)

### Evidencia 1

1. Tabla elegida.

TORNEO

2. Breve descripción de la restricción

La fecha de fin del torneo no puede ser antes de la fecha de inicio.

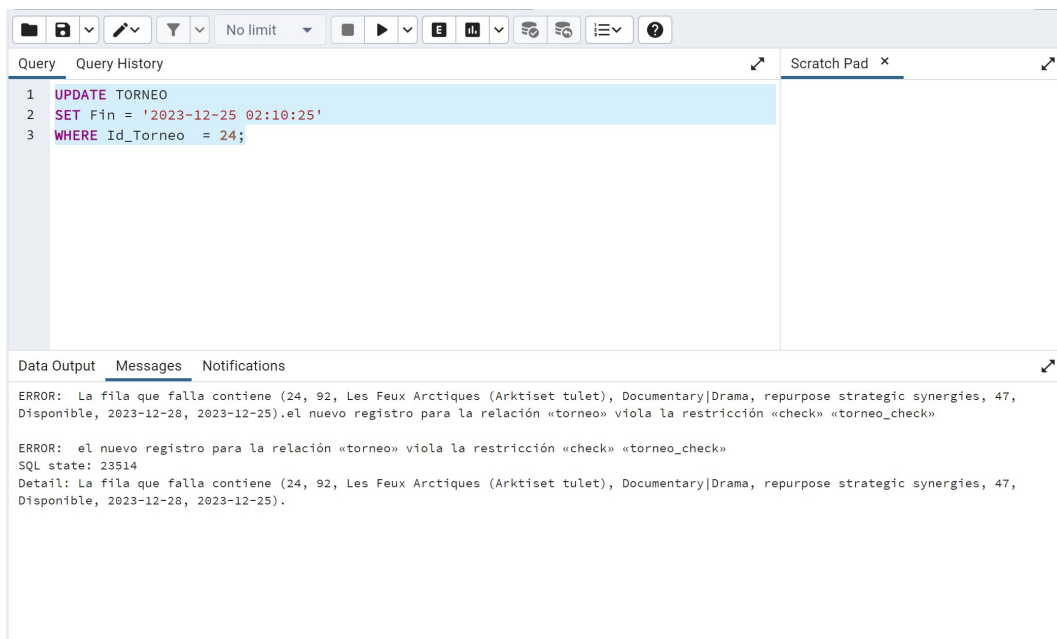
3. Instrucción para la creación de la restricción.

```
ALTER TABLE Torneo
ADD CONSTRAINT Fechas_validas
CHECK (Fin > Inicio);
```

4. Instrucción que permita evidenciar que la restricción esta funcionando.

```
UPDATE TORNEO
SET Fin = '2023-12-25 02:10:25'
WHERE Id_Torneo = 24;
```

5. Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.



### Evidencia 2

1. Tabla elegida.

AMISTAD

2. Breve descripción de la restricción

Un jugador no puede ser amigo de si mismo.

- Instrucción para la creación de la restricción.

```
ALTER TABLE Amistad
ADD CONSTRAINT Amigos_unicos
CHECK (Id_Jugador < Id_amigo);
```

- Instrucción que permita evidenciar que la restricción esta funcionando.

```
INSERT INTO AMISTAD (Id_Jugador, Id_amigo)
VALUES (50, 50);
```

- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.



## 8. Consultas

- Redacción clara de la consulta:  
Buscar torneos por el nombre del administrador que lo creo.
- Código en lenguaje SQL de la consulta.

```
SELECT Administrador.Nombre , Torneo.*
FROM Torneo
JOIN Administrador ON Torneo.ID_admin = Administrador.Id_Admin
WHERE Administrador.Nombre = 'Nombre del administrador';
```

- Ejecución y captura de pantalla.

|   | nombre<br>character varying (255) | id_torneo<br>integer | id_admin<br>integer | nombre<br>character varying (255)          | juego<br>character varying (255) | reglas<br>character varying (255)     | num_participantes<br>integer | estatus<br>character varying (16) | inicio<br>date | fin<br>date |
|---|-----------------------------------|----------------------|---------------------|--|----------------------------------|---------------------------------------|------------------------------|-----------------------------------|----------------|-------------|
| 1 | Jordain Bellon                    | 16                   | 12                  | The Story of Robin Hood and His Merrie Men | Action/Adventure/Children        | incubate cutting-edge infrastructures | 22                           | En curso                          | 2023-12-16     | 2024-01-19  |
| 2 | Jordain Bellon                    | 35                   | 12                  | Hudson Hawk                                | Action/Adventure/Comedy          | enhance best-of-breed relationships   | 43                           | Disponible                        | 2023-12-10     | 2024-01-13  |
| 3 | Jordain Bellon                    | 70                   | 12                  | Miss Bala                                  | Action/Adventure/Drama/Thriller  | brand user-centric convergence        | 47                           | Finalizado                        | 2023-12-28     | 2024-03-23  |
| 4 | Jordain Bellon                    | 95                   | 12                  | Home Run                                   | Drama                            | strategize ubiquitous action-items    | 11                           | En curso                          | 2023-12-11     | 2024-03-05  |

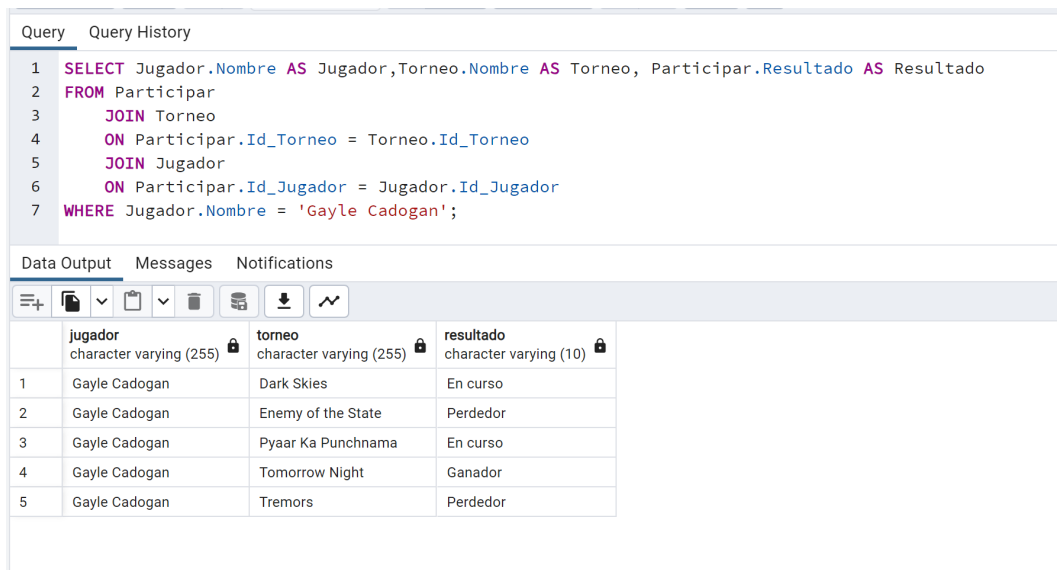
1. Redacción clara de la consulta:

Consultar torneos en los que el jugador ha participado o esta participando. La consulta se realiza con el nombre del jugador.

2. Código en lenguaje SQL de la consulta.

```
SELECT Jugador.Nombre AS Jugador,Torneo.Nombre AS Torneo, Participar.Resultado AS Resultado
FROM Participar
JOIN Torneo ON Participar.Id_Torneo = Torneo.Id_Torneo
JOIN Jugador ON Participar.Id_Jugador = Jugador.Id_Jugador
WHERE Jugador.Nombre = 'Nombre del jugador';
```

3. Ejecución y captura de pantalla.



The screenshot shows a SQL query editor with the following query:

```
1 SELECT Jugador.Nombre AS Jugador,Torneo.Nombre AS Torneo, Participar.Resultado AS Resultado
2 FROM Participar
3 JOIN Torneo
4 ON Participar.Id_Torneo = Torneo.Id_Torneo
5 JOIN Jugador
6 ON Participar.Id_Jugador = Jugador.Id_Jugador
7 WHERE Jugador.Nombre = 'Gayle Cadogan';
```

Below the query, the 'Data Output' tab is active, showing the results of the query in a table:

|   | jugador<br>character varying (255) | torneo<br>character varying (255) | resultado<br>character varying (10) |
|---|------------------------------------|-----------------------------------|-------------------------------------|
| 1 | Gayle Cadogan                      | Dark Skies                        | En curso                            |
| 2 | Gayle Cadogan                      | Enemy of the State                | Perdedor                            |
| 3 | Gayle Cadogan                      | Pyaar Ka Punchnama                | En curso                            |
| 4 | Gayle Cadogan                      | Tomorrow Night                    | Ganador                             |
| 5 | Gayle Cadogan                      | Tremors                           | Perdedor                            |

1. Redacción clara de la consulta:

Consultar la lista de amigos de un jugador, buscando por su nombre.

2. Código en lenguaje SQL de la consulta.

```
SELECT Amigo.Nombre AS Nombre_Amigo
FROM Amistad
JOIN Jugador ON Amistad.Id_Jugador = Jugador.Id_Jugador
JOIN Jugador Amigo ON Amistad.Id_amigo = Amigo.Id_Jugador
WHERE Jugador.Nombre = 'Winfred Dukesbury'
```

3. Ejecución y captura de pantalla.



The screenshot shows a SQL query editor with a 'Query' tab selected. The query is as follows:

```

1 SELECT Amigo.Nombre AS Nombre_Amigo
2 FROM Amistad
3 JOIN Jugador
4     ON Amistad.Id_Jugador = Jugador.Id_Jugador
5 JOIN Jugador Amigo
6     ON Amistad.Id_amigo = Amigo.Id_Jugador
7 WHERE Jugador.Nombre = 'Winfred Dukesbury'
8
9

```

Below the query editor, there is a 'Data Output' tab showing the results of the query. The results are as follows:

|   | nombre_amigo<br>character varying (255) |
|---|---|
| 1 | Tuesday Falconar                        |
| 2 | Franklyn Aykroyd                        |
| 3 | Emmit Infantino                         |
| 4 | Cobby Grishagin                         |

## 9. Vistas

- Redacción clara de la vista planteada. Historial de los participantes de cada torneo. La vista debe mostrar el listado de los participantes de cada torneo. Estarán ordenados alfabéticamente por el nombre de los torneos y debe mostrar el nombre del jugador, si gana o perdio, su correo y su nivel actual. La vista debe incluir los siguientes campos:

|               |                |           |                |       |
|---------------|----------------|-----------|----------------|-------|
| nombre_torneo | nombre_jugador | resultado | correo_jugador | nivel |
|---------------|----------------|-----------|----------------|-------|

- Código en lenguaje SQL que permita crear la vista solicitada.

```

CREATE VIEW Vista_Participantes_Torneo AS
SELECT
Torneo.Nombre AS Nombre_Torneo,
Jugador.Nombre AS Nombre_Jugador,
Participar.Resultado,
Jugador.Correo AS Correo_Jugador,
Jugador.Nivel
FROM Participar
JOIN Torneo ON Participar.Id_Torneo = Torneo.Id_Torneo
JOIN Jugador ON Participar.Id_Jugador = Jugador.Id_Jugador
ORDER BY Torneo.Nombre, Jugador.Nombre;

```

- Ejecutar el código para la creación de la vista en Postgres e incluir una captura de pantalla con la vista creada satisfactoriamente.

Query    Query History

```
1 CREATE VIEW Vista_Participantes_Torneo AS
2 SELECT
3     Torneo.Nombre AS Nombre_Torneo,
4     Jugador.Nombre AS Nombre_Jugador,
5     Participar.Resultado,
6     Jugador.Correo AS Correo_Jugador,
7     Jugador.Nivel
8 FROM Participar
9 JOIN Torneo ON Participar.Id_Torneo = Torneo.Id_Torneo
10 JOIN Jugador ON Participar.Id_Jugador = Jugador.Id_Jugador
11 ORDER BY Torneo.Nombre, Jugador.Nombre;
12
13
14 SELECT * FROM Vista_Participantes_Torneo;
```

Data Output    Messages    Notifications

|    | nombre_torneo<br>character varying (255)          | nombre_jugador<br>character varying (255) | resultado<br>character varying (10) | correo_jugador<br>character varying (255) | nivel<br>integer |
|----|---|---|-------------------------------------|---|------------------|
| 1  | 30 Days of Night: Dark Days                       | Yolanthe Joynson                          | Perdedor                            | yjoynson1w@desdev.cn                      | 22               |
| 2  | Angela's Ashes                                    | Joshuah Lebel                             | Perdedor                            | jlebel1z@wikipedia.org                    | 47               |
| 3  | Angela's Ashes                                    | Raynard Jelly                             | Ganador                             | ryelly1f@etsy.com                         | 76               |
| 4  | Antz  | Gardiner Aberkirdo                        | Perdedor                            | gaberkirdox@gov.uk                        | 48               |
| 5  | Applesseed (Appurushido)                          | Hilda Pilbury                             | En curso                            | hpilburyv@apple.com                       | 1                |
| 6  | Arabesque   | Manfred Catton                            | Ganador                             | mcatton5@forbes.com                       | 57               |
| 7  | At Play in the Fields of the Lord                 | Almeria Christoforou                      | Ganador                             | achristoforouj@smh.com.au                 | 74               |
| 8  | At Play in the Fields of the Lord                 | Bruis Martinson                           | Ganador                             | bmartinson18@cafepress.com                | 66               |
| 9  | At Play in the Fields of the Lord                 | Reece Swin                                | En curso                            | rswin0@infoseek.co.jp                     | 63               |
| 10 | Berserk: The Golden Age Arc - The Egg of the King | Elissa McGarahan                          | Ganador                             | emcgarahan24@nydailynews...               | 43               |
| 11 | Berserk: The Golden Age Arc - The Egg of the King | Kalina Milazzo                            | Ganador                             | kmilazzo2n@patch.com                      | 13               |
| 12 | Better Tomorrow III: Love and Death in Saigon, A  | Sherwin Medeway                           | Perdedor                            | smedewayu@imageshack.us                   | 91               |
| 13 | Bo Burnham: Words, Words, Words                   | Agustin Sygroves                          | Perdedor                            | asygrovesp@eepurl.com                     | 13               |
| 14 | Bo Burnham: Words, Words, Words                   | Norry Melbury                             | En curso                            | nmelbury@ezinearticles.com                | 84               |
| 15 | Buck  | Cleon Boswell                             | Ganador                             | cboswell9@w3.org                          | 52               |

1. Redacción clara de la vista planteada. Historial de los participantes que han ganado algún torneo, así como la cantidad de torneos ganados, de esta forma se podrá tener un ranking al estar ordenados de manera decreciente.
- La vista debe incluir los siguientes campos:

|            |                |                          |
|------------|----------------|--------------------------|
| id_jugador | nombre_jugador | cantidad_torneos_ganados |
|------------|----------------|--------------------------|

2. Código en lenguaje SQL que permita crear la vista solicitada.

```
CREATE VIEW Vista_Ganadores AS
SELECT
    Jugador.Id_Jugador,
    Jugador.Nombre AS Nombre_Jugador,
    COUNT(Participar.Id_Torneo) AS Cantidad_Torneos_Ganados
FROM Participar
JOIN Jugador ON Participar.Id_Jugador = Jugador.Id_Jugador
WHERE Participar.Resultado = 'Ganador'
GROUP BY Jugador.Id_Jugador, Jugador.Nombre
ORDER BY Cantidad_Torneos_Ganados DESC;
```

3. Ejecutar el código para la creación de la vista en Postgres e incluir una captura de pantalla con la vista creada satisfactoriamente.

| Query   |                         | Query History            |
|---|-------------------------|--------------------------|
| <pre> 1 CREATE VIEW Vista_Ganadores AS 2 SELECT 3     Jugador.Id_Jugador, 4     Jugador.Nombre AS Nombre_Jugador, 5     COUNT(Participar.Id_Torneo) AS Cantidad_Torneos_Ganados 6 FROM Participar 7 JOIN Jugador ON Participar.Id_Jugador = Jugador.Id_Jugador 8 WHERE Participar.Resultado = 'Ganador' 9 GROUP BY Jugador.Id_Jugador, Jugador.Nombre 10 ORDER BY Cantidad_Torneos_Ganados DESC; 11 12 SELECT * FROM Vista_Ganadores; 13 </pre> |                         |                          |
| Data Output   |                         | Messages                 |
| Notifications   |                         |                          |
| id_jugador  | nombre_jugador          | cantidad_torneos_ganados |
| integer   | character varying (255) | bigint                   |
| 1   | 45                      | Bruis Martinson          |
| 2   | 72                      | Joshuah Lebel            |
| 3   | 77                      | Elissa McGarahan         |
| 4   | 64                      | Dan Heaselgrave          |
| 5   | 10                      | Cleon Boswell            |
| 6   | 38                      | Franny Teck              |
| 7   | 6                       | Manfred Catton           |
| 8   | 26                      | Agustin Sygroves         |
| 9   | 25                      | Jayne Barthelmes         |
| 10  | 50                      | Laverne Marco            |
| 11  | 21                      | Haley Peach              |
| 12  | 97                      | Breana Bootes            |
| 13  | 20                      | Almeria Christoforou     |
| 14  | 22                      | Tabb Hardington          |
| 15  | 74                      | Gayle Cadogan            |
| 16  | 2                       | Culver MacKintosh        |
| 17  | 29                      | Falito Bewlie            |

- Redacción clara de la vista planteada. Ranking de los jugadores mas populares, la vista mostrara en orden decreciente los jugadores y la cantidad de amigos que tienen, mostrando al principio a los jugadores con mas amigos.

La vista debe incluir los siguientes campos:

|            |                |                 |
|------------|----------------|-----------------|
| id_jugador | nombre_jugador | cantidad_amigos |
|------------|----------------|-----------------|

- Código en lenguaje SQL que permita crear la vista solicitada.

```

CREATE VIEW Vista_Cantidad_Amigos AS
SELECT
Jugador.Id_Jugador,
Jugador.Nombre AS Nombre_Jugador,
COUNT(Amistad.Id_amigo) AS Cantidad_Amigos
FROM Jugador
LEFT JOIN Amistad ON Jugador.Id_Jugador = Amistad.Id_Jugador
GROUP BY Jugador.Id_Jugador, Jugador.Nombre
ORDER BY Cantidad_Amigos DESC;

```

- Ejecutar el código para la creación de la vista en Postgres e incluir una captura de pantalla con la vista creada satisfactoriamente.

Query Query History

```
1 CREATE VIEW Vista_Cantidad_Amigos AS
2 SELECT
3     Jugador.Id_Jugador,
4     Jugador.Nombre AS Nombre_Jugador,
5     COUNT(Amistad.Id_amigo) AS Cantidad_Amigos
6 FROM Jugador
7 LEFT JOIN Amistad ON Jugador.Id_Jugador = Amistad.Id_Jugador
8 GROUP BY Jugador.Id_Jugador, Jugador.Nombre
9 ORDER BY Cantidad_Amigos DESC;
10
11
12 SELECT * FROM Vista_Cantidad_Amigos;
```

Data Output Messages Notifications

|    | id_jugador<br>integer | nombre_jugador<br>character varying (255) | cantidad_amigos<br>bigint |
|----|-----------------------|---|---------------------------|
| 1  | 70                    | Arie Blunn                                | 4                         |
| 2  | 99                    | Winfred Dukesbury                         | 4                         |
| 3  | 97                    | Breena Bootes                             | 3                         |
| 4  | 1                     | Reece Swin                                | 3                         |
| 5  | 50                    | Laverne Marco                             | 3                         |
| 6  | 13                    | Waite MacTague                            | 3                         |
| 7  | 7                     | Marylinda Bohlje                          | 2                         |
| 8  | 28                    | Merilee Llewellen                         | 2                         |
| 9  | 43                    | Lanita Thurber                            | 2                         |
| 10 | 21                    | Haley Peach                               | 2                         |
| 11 | 46                    | Luisa Muckeen                             | 2                         |
| 12 | 3                     | Chelsea Ruegg                             | 2                         |
| 13 | 83                    | Waly Woodlands                            | 2                         |
| 14 | 56                    | Gardiner Heitz                            | 2                         |
| 15 | 59                    | Roderigo Purkess                          | 2                         |