

## 2.2. Ejercicio 1. Diseño conceptual de la base de datos para la gestión de los Juegos Olímpicos (20%)

Nos solicitan:

- 1) realizar una revisión del modelo conceptual completo facilitado con el objetivo de indicar si la representación de información es correcta o no en base a los requerimientos descritos.
- 2) Para cada error u omisión detectados, modificar el diseño conceptual para corregirlo.

### **Cambios realizados:**

#### **1. BELONGS TO COUNTRY entre TEAM y COUNTRY:**

- **Antes:** Se interpretaba que cada país tenía solo un equipo y que un equipo podía pertenecer a varios países, lo cual es incorrecto
- **Cambio:** Ahora, se ha establecido la cardinalidad (N,1) para esta relación, donde:
  - N (múltiple) para TEAM: Un país puede tener múltiples equipos, como en el caso de diferentes equipos de natación, atletismo, baloncesto, etc., que representan al mismo país.
  - 1 (uno) para COUNTRY: Cada equipo pertenece a un único país.
- **Resultado:** Ahora se refleja correctamente que un país puede tener varios equipos, pero cada equipo está asociado a un solo país.

#### **2. IS LOCATED IN entre CITY y COUNTRY:**

- **Antes:** Cada ciudad solo estaba en un país, pero no reflejaba que un país pudiera tener varias ciudades.
- **Cambio:** Se ajustó a la cardinalidad (N,1), donde:
  - N (múltiple) para CITY: Un país puede tener múltiples ciudades.
  - 1 (uno) para COUNTRY: Cada ciudad se encuentra en un único país.
- **Resultado:** Esto asegura que cada ciudad esté ubicada en un solo país, pero permite que un país tenga varias ciudades, reflejando mejor la estructura geográfica.

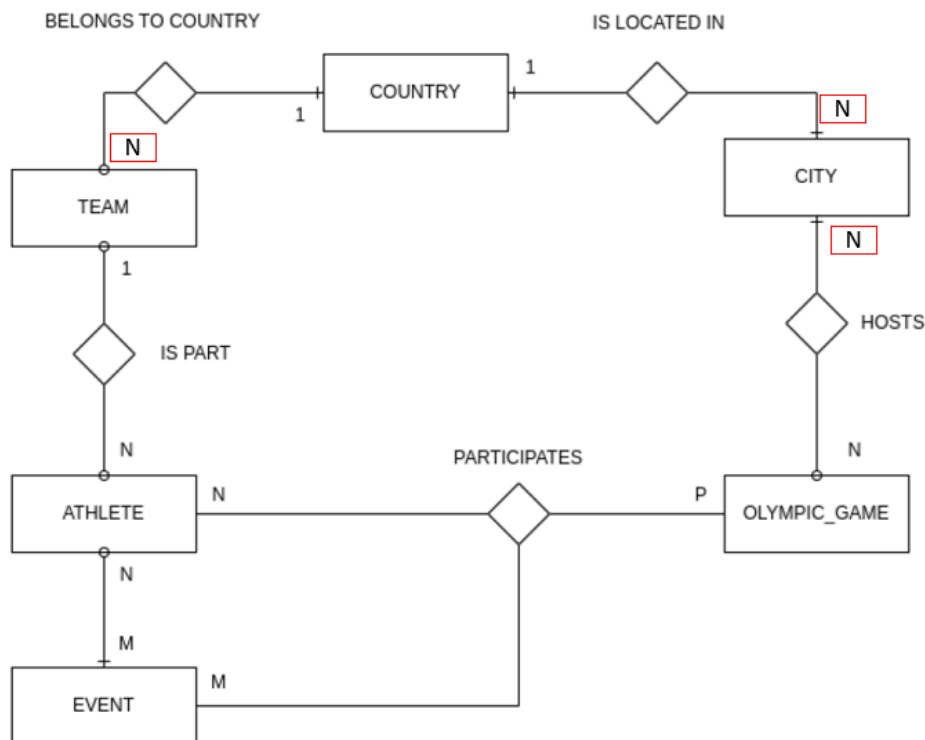
#### **3. Se ha identificado que falta la entidad Deporte, que agrupe los eventos deportivos. Pero como es solicitado en el ejercicio 2.2.2, no se procede a la modificación**

#### **4. HOSTS entre CITY y OLYMPIC GAME:**

- **Antes:** Se interpreta que cada ciudad solo podía albergar una edición de los Juegos.
- **Cambio:** Se modificó la cardinalidad como (N,1), donde:

- **N** (múltiple) para **CITY**: Una ciudad puede ser anfitriona en múltiples ediciones de los Juegos Olímpicos.
- **1** (uno) para **OLYMPIC\_GAME**: Cada edición de los Juegos se celebra en una única ciudad.
- **Resultado**: Esto permite que una misma ciudad pueda ser anfitriona de varias ediciones de los Juegos (por ejemplo, París en 1900, 1924, y 2024), mientras que cada edición específica se realiza en una sola ciudad.

### Mapa conceptual modificado



### 2.2.2. Evolución del modelo conceptual E/R

Una vez revisado el modelo conceptual y corregidos los errores, nos piden que soporte para que la aplicación pueda incorporar otra dos nuevas funcionalidades.

Se desea poder llevar el control de las diferentes disciplinas deportivas, las cuales agruparán los diferentes eventos o acontecimientos deportivos en los que están especializados los atletas. Se desea conocer para cada uno de los acontecimientos deportivos a qué disciplina deportiva pertenece, siendo esta, obligatoria, por lo que se almacenará en la base de datos para cada disciplina un código numérico de disciplina, que es su identificador y una descripción.

Y por otro lado se desea poder conocer en cada participación de un atleta en un evento, en caso de haberse clasificado entre los tres primeros, qué medalla ha conseguido.

Por ello se solicita adaptar el modelo conceptual de datos como mejor convenga para

poder llevar a cabo esta nueva funcionalidad. En concreto se solicita:

1) La modificación del diseño conceptual justificando cada cambio a realizar.

#### Nuevas entidades:

- **DISCIPLINE**  
discipline\_code, description
- **PARTICIPATION** (entidad débil de ATHLETE, OLYMPIC\_GAME, y EVENT)  
Medal type

#### Detalles de PARTICIPATION como Entidad Débil

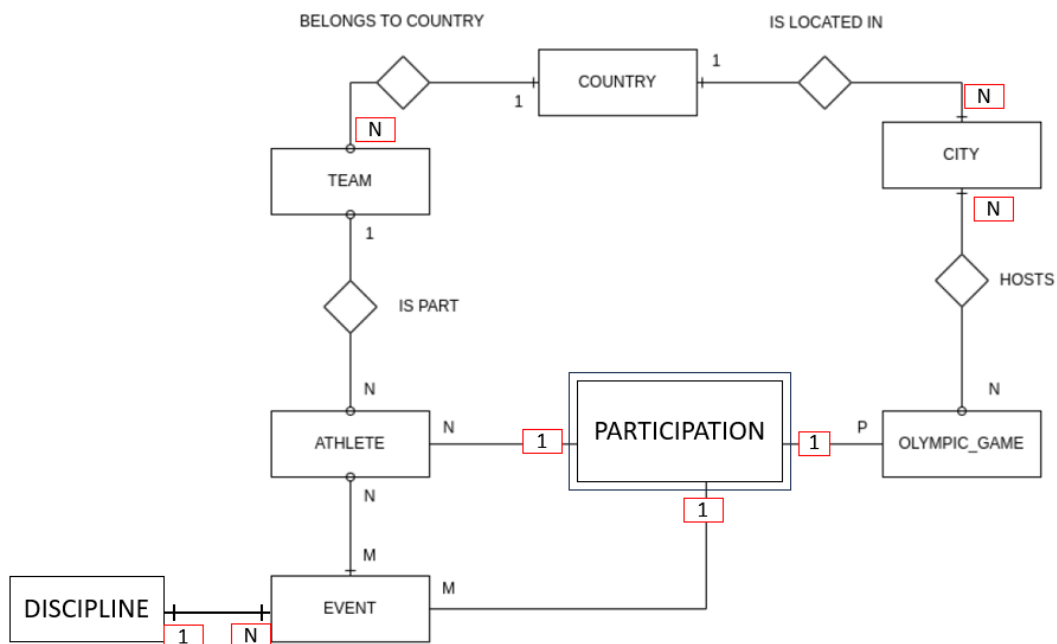
##### 1. Dependencia Existencial:

- **PARTICIPATION** no tiene una clave primaria propia que la identifique de manera independiente. Su existencia y significado dependen de los tres elementos: **ATHLETE**, **OLYMPIC\_GAME**, y **EVENT**.
- Por lo tanto, **PARTICIPATION** es una entidad débil en relación con estas tres entidades, ya que no puede existir sin una combinación específica de ellas.

##### 2. Clave Parcial de PARTICIPATION:

- La clave primaria de **PARTICIPATION** se forma utilizando las claves primarias de **ATHLETE**, **OLYMPIC\_GAME**, y **EVENT** como clave compuesta.
- La combinación de estas tres claves define cada instancia única de **PARTICIPATION**, indicando que un atleta participó en un evento específico en una edición específica de los Juegos Olímpicos.

Además, el atributo **medal\_type** (para reflejar la medalla obtenida en cada participación) depende directamente de esta combinación y no tiene sentido fuera de ella.



### 2.3. Ejercicio 2. Diseño lógico de la base de datos para la gestión de los Juegos Olímpicos (10%)

Llevadas a cabo las correcciones indicadas en el apartado 2.2. Ejercicio 1 e incluidos los cambios solicitados en el apartado 2.3. Ejercicio 2, se solicita:

- 1) Identificar las modificaciones a realizar en el modelo lógico propuesto para incorporar las correcciones y cambios solicitados.
- 2) Expresar el modelo lógico final utilizando la notación empleada en los módulos didácticos. Deben de indicarse claves primarias, alternativas, si existen, y claves foráneas.

**OLYMPIC\_GAME** (olympic\_game\_code, year, season)

**ATHLETE** (athlete\_code, name, sex, age, height, weight, team\_code)

donde {team\_code} referencia TEAM (team\_code)

**COUNTRY** (country\_code, country\_name)

**TEAM** (team\_code, full\_name, country\_code)

donde {country\_code} referencia COUNTRY (country\_code)

**CITY** (city\_code, full\_name, country\_code)

donde {country\_code} referencia COUNTRY (country\_code)

**EVENT** (event\_code, event\_description, discipline\_code)

y {discipline\_code} referencia DISCIPLINE (discipline\_code)

**DISCIPLINE** (discipline\_code, discipline\_description)

**HOSTS** (olympic\_game\_code, city\_code)

donde {olympic\_game\_code} referencia OLYMPIC\_GAME (olympic\_game\_code) y {city\_code} referencia CITY (city\_code)

**PARTICIPATION** (athlete\_code, event\_code, olympic\_game\_code, medal\_type)

{athlete\_code} referencia ATHLETE (athlete\_code)

{event\_code} referencia EVENT (event\_code)

{olympic\_game\_code} referencia OLYMPIC\_GAME (olympic\_game\_code)

{medal\_type} puede tomar valor nulo (Oro, Plata, Bronce)

2.4. Ejercicio 3. Modelo físico de la base de datos para la gestión de los Juegos Olímpicos (30%)

Ejecución de query

pgAdmin 4

File Object Tools Help

Object Explorer

SOURCE\_conxigb

SOURCE\_cventuralo

SOURCE\_danidvo

CastsCatalogsEvent TriggersExtensionsForeign Data WrappersLanguagesPublicationsSchemas (2)dboAggregatesCollationsDomainsFTS ConfigurationsFTS DictionariesFTS ParsersFTS TemplatesForeign TablesFunctionsMaterialized ViewsOperatorsProceduresSequencesTables (4)athleteeventspecializedsportTrigger FunctionsTypesViews

SOURCE\_danidvo/STUDENT\_danidvo@PostgreSQL\_UOC\*

SOURCE\_danidvo/STUDENT\_danidvo@PostgreSQL\_UOC

No limit

Query

Query History

1 SELECT

2 count(1) as sportCount

3 FROM

4 dbo.sport

Data Output

Messages

Notifications

+

+

+

+

+

+

+

+

sportcount

bigint

1

66

SOURCE\_danidvo/STUDENT\_danidvo@PostgreSQL\_UOC\* X

SOURCE\_danidvo/STUDENT\_danidvo@PostgreSQL\_UOC

No limit

?

Query Query History

```
1 SELECT *
2 FROM
3 dbo.sport
4 WHERE
5 sport_description = 'Athletics'
```

Data Output Messages Notifications

	sport_code integer	sport_description character varying (100)
1	6	Athletics

SOURCE\_danidvo/STUDENT\_danidvo@PostgreSQL\_UOC\* X

SOURCE\_danidvo/STUDENT\_danidvo@PostgreSQL\_UOC

No limit

?

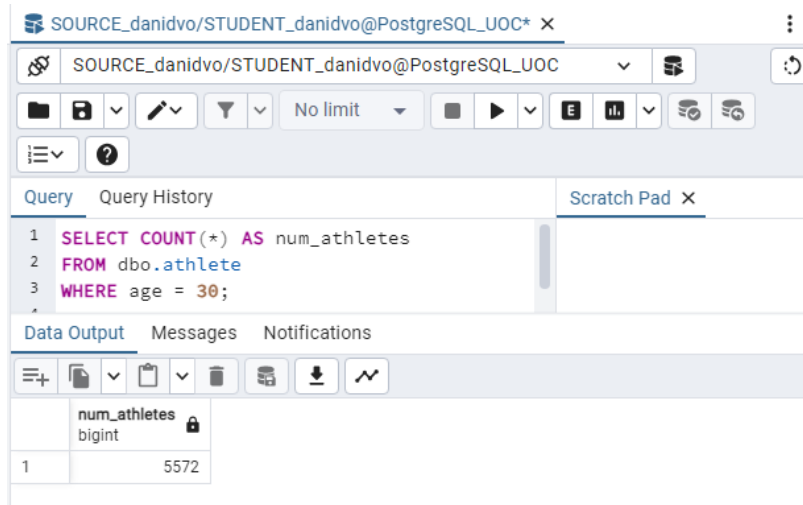
Query Query History

```
1 SELECT *
2 FROM
3 dbo.sport
4 WHERE
5 sport_description like 'T%'
```

Data Output Messages Notifications

	sport_code integer	sport_description character varying (100)
1	57	Table Tennis
2	58	Taekwondo
3	59	Tennis
4	60	Trampolining
5	61	Triathlon
6	62	Tug-Of-War

- 1) ¿Cuántos atletas de la base de datos tienen exactamente 30 años?



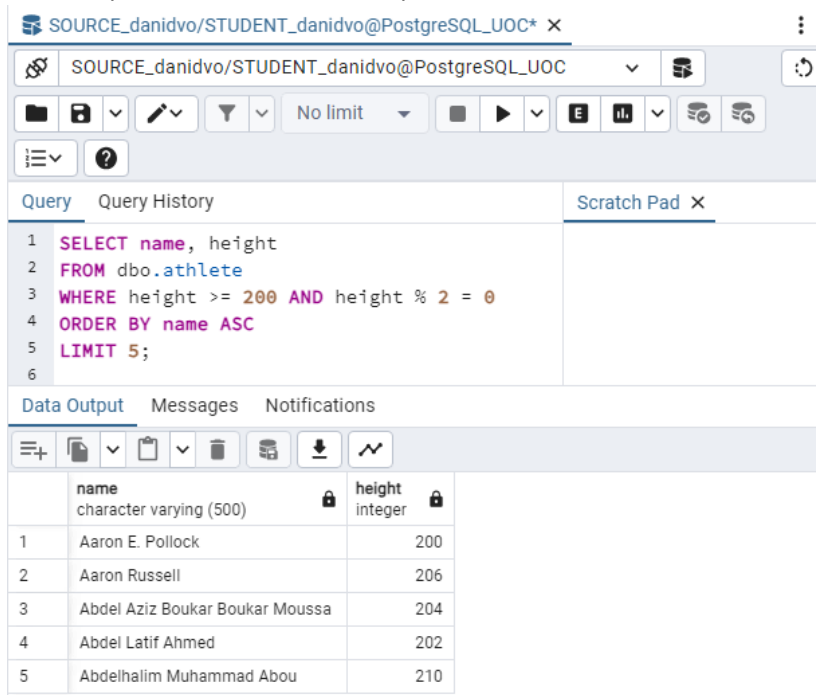
The screenshot shows a PostgreSQL query editor interface. The query is as follows:

```
1 SELECT COUNT(*) AS num_athletes
2 FROM dbo.athlete
3 WHERE age = 30;
```

The 'Data Output' tab is active, displaying the result of the query:

	num_athletes bigint
1	5572

- 2) Listar los 5 primeros atletas cuya altura sea mayor o igual a 200 cm, cuya altura sea un número par, ordenando la salida por nombre de contenido ascendente.



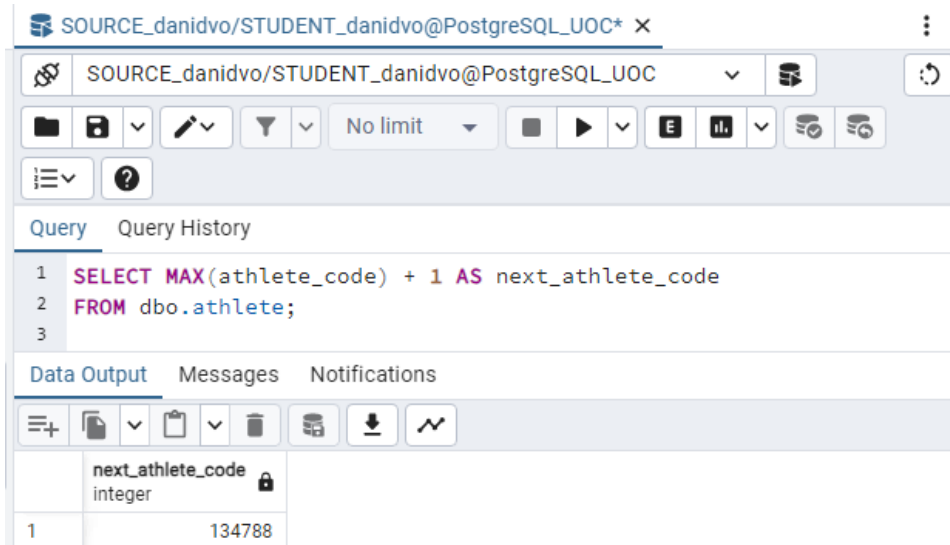
The screenshot shows a PostgreSQL query editor interface. The query is as follows:

```
1 SELECT name, height
2 FROM dbo.athlete
3 WHERE height >= 200 AND height % 2 = 0
4 ORDER BY name ASC
5 LIMIT 5;
```

The 'Data Output' tab is active, displaying the result of the query:

	name character varying (500)	height integer
1	Aaron E. Pollock	200
2	Aaron Russell	206
3	Abdel Aziz Boukar Boukar Moussa	204
4	Abdel Latif Ahmed	202
5	Abdelhalim Muhammad Abou	210

- 3) Insertar en la tabla athlete un nuevo registro teniendo en cuenta que:
- el valor del campo athlete\_code se informará con el valor numérico siguiente que corresponda. Para conocer el último valor existente en la clave primaria (athlete\_code), se deberá de realizar una consulta previa para conocer el valor máximo en este campo.



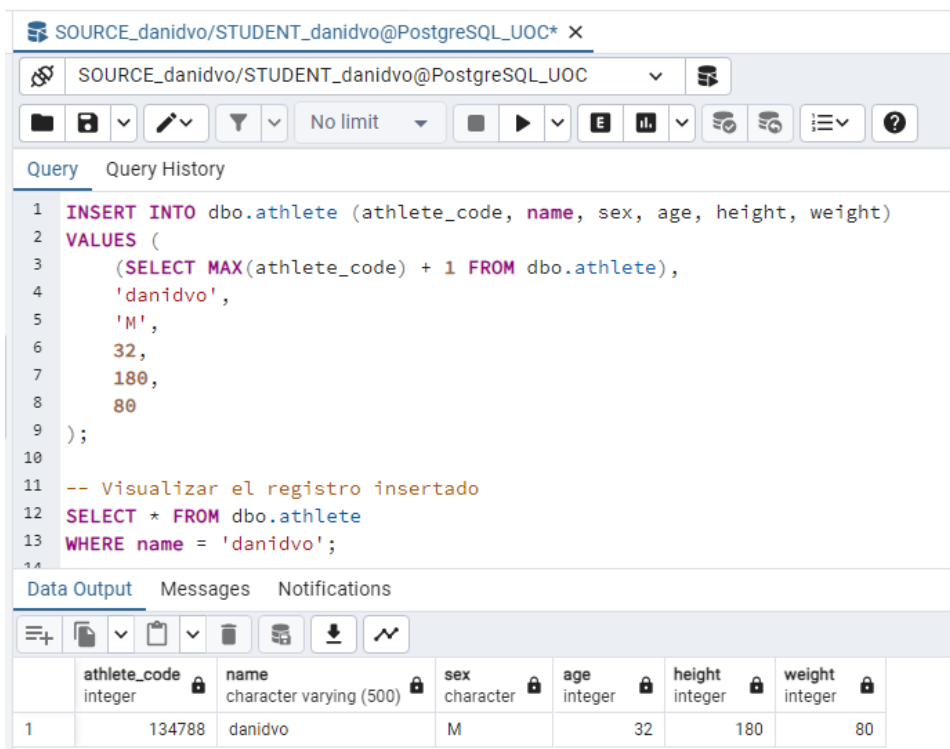
The screenshot shows a PostgreSQL query editor interface. The query is as follows:

```
1 SELECT MAX(athlete_code) + 1 AS next_athlete_code
2 FROM dbo.athlete;
3
```

The 'Data Output' tab is selected, showing the result of the query:

	next_athlete_code integer
1	134788

- b) el valor del campo athlete\_name se informará con vuestro 'loginuoc' asignado en el campus, los campos sex, age, height y weight se informarán con valores libres que se consideren. Una vez insertado el registro, visualizarlo mediante una consulta.



The screenshot shows a PostgreSQL query editor interface. The query is as follows:

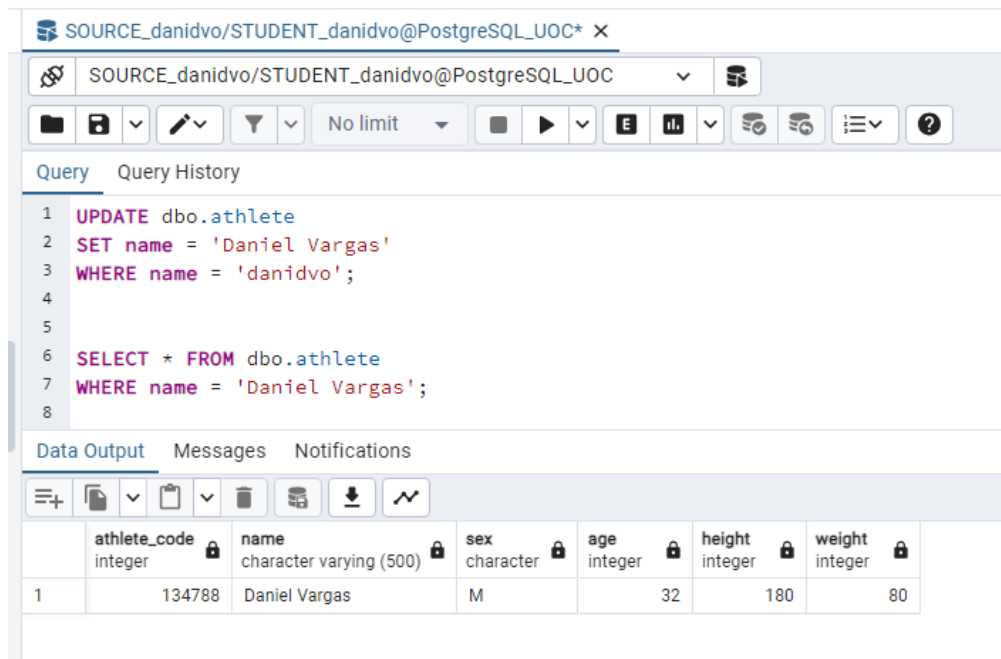
```
1 INSERT INTO dbo.athlete (athlete_code, name, sex, age, height, weight)
2 VALUES (
3     (SELECT MAX(athlete_code) + 1 FROM dbo.athlete),
4     'danidvo',
5     'M',
6     32,
7     180,
8     80
9 );
10
11 -- Visualizar el registro insertado
12 SELECT * FROM dbo.athlete
13 WHERE name = 'danidvo';
14
```

The 'Data Output' tab is selected, showing the result of the query:

	athlete_code integer	name character varying (500)	sex character	age integer	height integer	weight integer
1	134788	danidvo	M	32	180	80

- 4) Actualizar el registro insertado en el punto anterior, sustituir el valor de vuestro 'loginuoc' asignado en el campus por vuestro nombre completo (nombre y apellidos) y consultar el registro para visualizarlo.





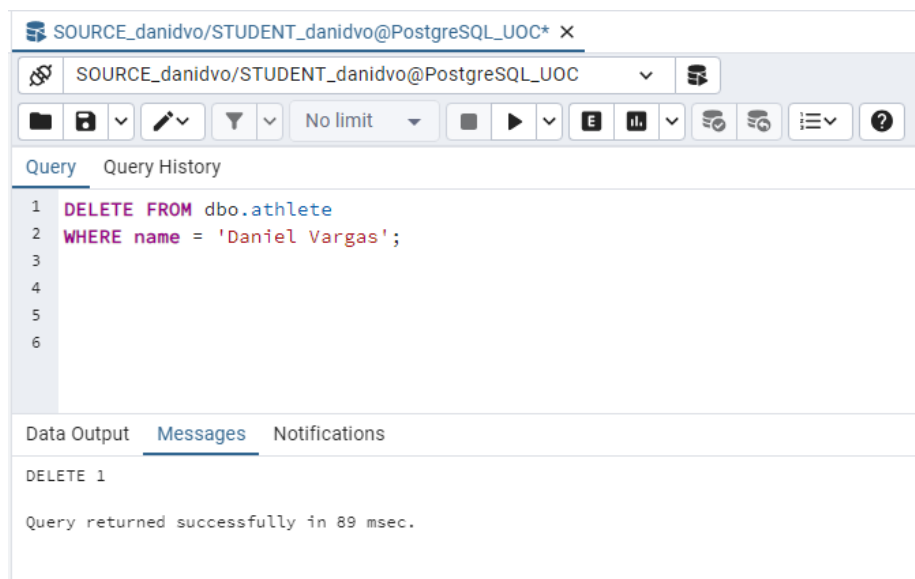
The screenshot shows a PostgreSQL query editor interface. The query window contains the following SQL code:

```
1 UPDATE dbo.athlete
2 SET name = 'Daniel Vargas'
3 WHERE name = 'danidvo';
4
5
6 SELECT * FROM dbo.athlete
7 WHERE name = 'Daniel Vargas';
8
```

Below the query window, the 'Data Output' tab is active, displaying a table with the following data:

	athlete_code integer	name character varying (500)	sex character	age integer	height integer	weight integer
1	134788	Daniel Vargas	M	32	180	80

5) Eliminar el nuevo registro insertado en el punto 3) y actualizado en el punto 4)



The screenshot shows a PostgreSQL query editor interface. The query window contains the following SQL code:

```
1 DELETE FROM dbo.athlete
2 WHERE name = 'Daniel Vargas';
3
4
5
6
```

Below the query window, the 'Messages' tab is active, displaying the following output:

```
DELETE 1
Query returned successfully in 89 msec.
```

6) Listar los eventos deportivos que constituyen el deporte “Athletics” ordenado ascendentemente por el nombre del evento.

SOURCE\_danidvo/STUDENT\_danidvo@PostgreSQL\_UOC\* X

SOURCE\_danidvo/STUDENT\_danidvo@PostgreSQL\_UOC

No limit

Query Query History

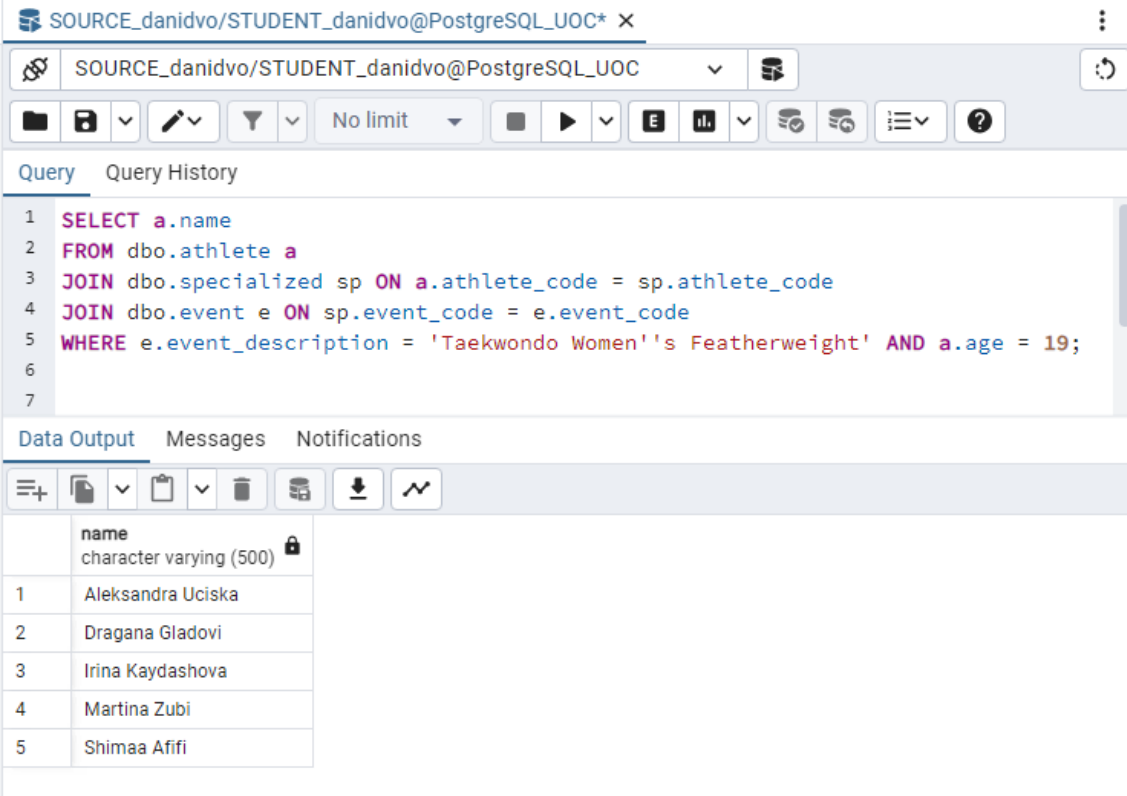
```
1 SELECT e.event_description
2 FROM dbo.event e
3 JOIN dbo.sport s ON e.sport_code = s.sport_code
4 WHERE s.sport_description = 'Athletics'
5 ORDER BY e.event_description ASC;
6
```

Data Output Messages Notifications

	event_description character varying (500)
1	Athletics Men's 1,500 metres
2	Athletics Men's 1,500 metres Walk
3	Athletics Men's 1,600 metres Medley Relay
4	Athletics Men's 10 kilometres Walk
5	Athletics Men's 10 mile Walk
6	Athletics Men's 10,000 metres
7	Athletics Men's 100 metres
8	Athletics Men's 110 metres Hurdles
9	Athletics Men's 2,500 metres Steeplechase
10	Athletics Men's 2,590 metres Steeplechase
11	Athletics Men's 20 kilometres Walk
12	Athletics Men's 200 metres
13	Athletics Men's 200 metres Hurdles
14	Athletics Men's 3 mile, Team
15	Athletics Men's 3,000 metres Steeplechase
16	Athletics Men's 3,000 metres Walk
17	Athletics Men's 3,000 metres, Team
18	Athletics Men's 3,200 metres Steeplechase
19	Athletics Men's 3,500 metres Walk
20	Athletics Men's 4 mile, Team
21	Athletics Men's 4 x 100 metres Relay

Total rows: 83 of 83 Query complete 00:00:00.093 Ln 5, Col 34

7) Listar las atletas especializadas en “Taekwondo Women's Featherweight” y con 19 años de edad.



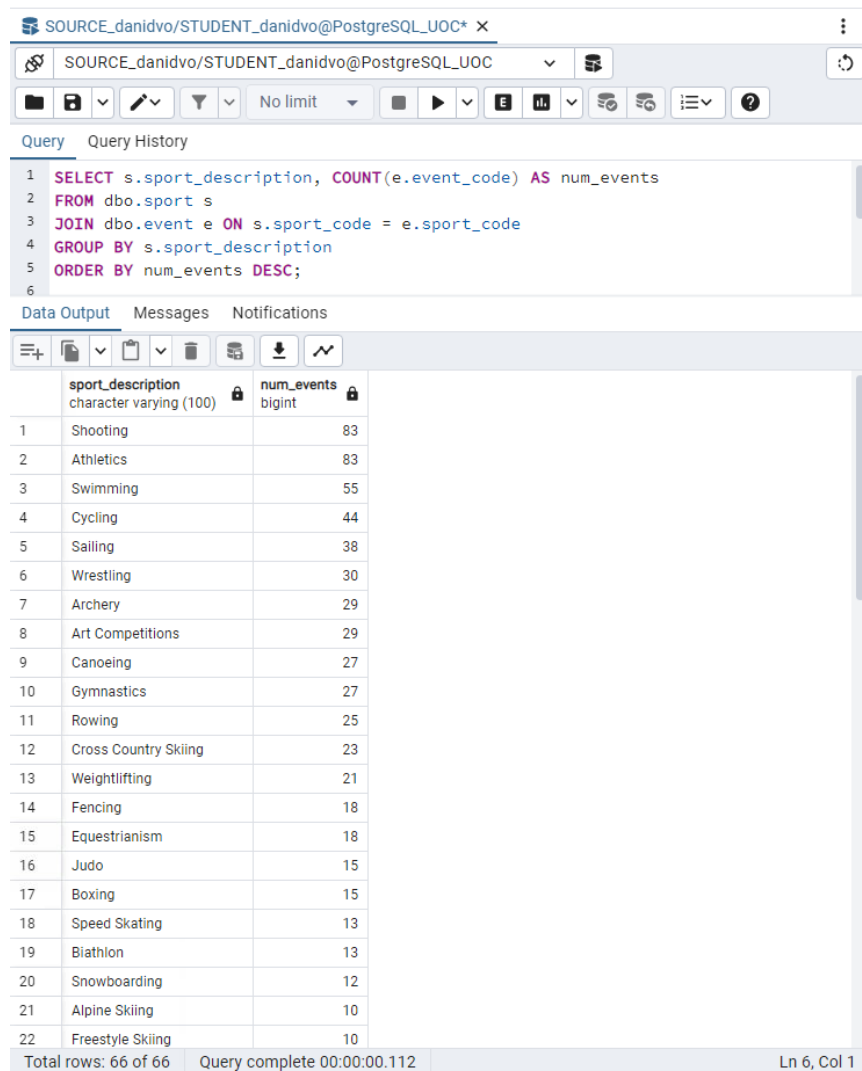
The screenshot shows a PostgreSQL query editor interface. The title bar indicates the connection is 'SOURCE\_danidvo/STUDENT\_danidvo@PostgreSQL\_UOC'. The query editor contains the following SQL query:

```
1 SELECT a.name
2 FROM dbo.athlete a
3 JOIN dbo.specialized sp ON a.athlete_code = sp.athlete_code
4 JOIN dbo.event e ON sp.event_code = e.event_code
5 WHERE e.event_description = 'Taekwondo Women's Featherweight' AND a.age = 19;
6
7
```

Below the query editor, the 'Data Output' tab is active, displaying the results of the query in a table format. The table has one column, 'name', with a data type of 'character varying (500)'. The results show five athletes:

	name
1	Aleksandra Uciska
2	Dragana Gladovi
3	Irina Kaydashova
4	Martina Zubi
5	Shimaa Afifi

8) Listar el número de eventos que constituyen cada deporte, ordenado de forma descendente.



The screenshot shows a PostgreSQL query editor interface. The query is as follows:

```
1 SELECT s.sport_description, COUNT(e.event_code) AS num_events
2 FROM dbo.sport s
3 JOIN dbo.event e ON s.sport_code = e.sport_code
4 GROUP BY s.sport_description
5 ORDER BY num_events DESC;
```

The results are displayed in a table with two columns: **sport\_description** (character varying (100)) and **num\_events** (bigint). The results are ordered by the number of events in descending order.

	sport_description	num_events
1	Shooting	83
2	Athletics	83
3	Swimming	55
4	Cycling	44
5	Sailing	38
6	Wrestling	30
7	Archery	29
8	Art Competitions	29
9	Canoeing	27
10	Gymnastics	27
11	Rowing	25
12	Cross Country Skiing	23
13	Weightlifting	21
14	Fencing	18
15	Equestrianism	18
16	Judo	15
17	Boxing	15
18	Speed Skating	13
19	Biathlon	13
20	Snowboarding	12
21	Alpine Skiing	10
22	Freestyle Skiing	10

Total rows: 66 of 66    Query complete 00:00:00.112    Ln 6, Col 1

9) Listar el nombre de los atletas y por cada uno de ellos el número de especializaciones deportivas que tienen, para aquellos que tienen un número de especializaciones mayor a 20, y ordenándolos de forma descendente por el número de especializaciones deportivas.

SOURCE\_danidvo/STUDENT\_danidvo@PostgreSQL\_UOC\* x

SOURCE\_danidvo/STUDENT\_danidvo@PostgreSQL\_UOC

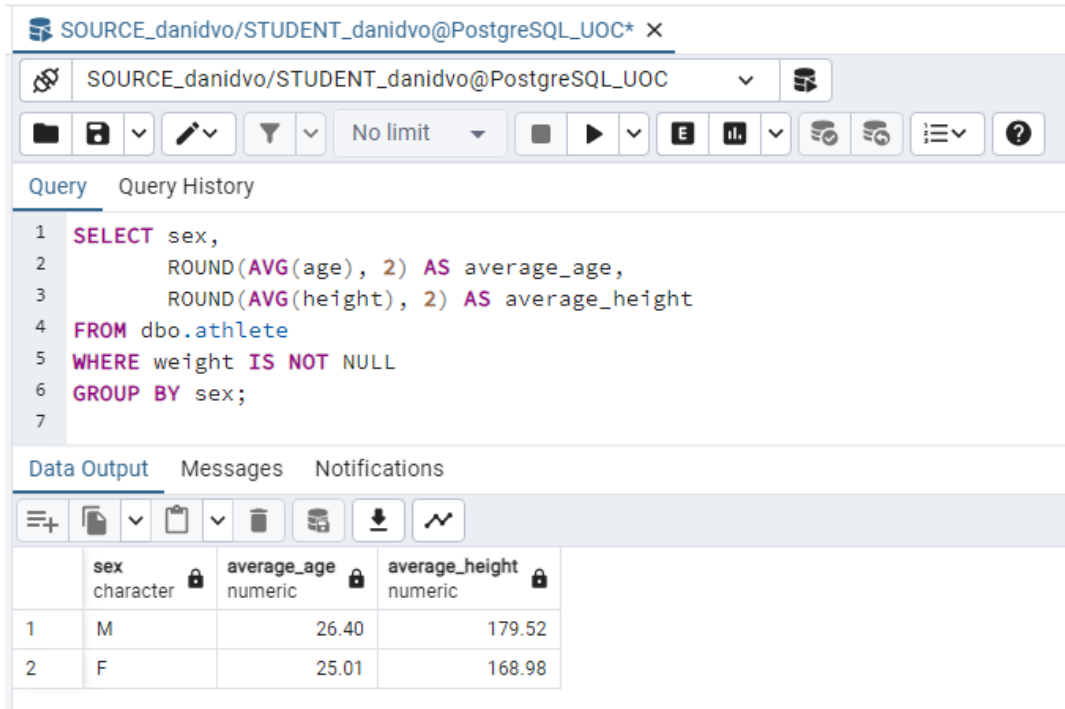
Query Query History

```
1 SELECT a.name, COUNT(sp.event_code) AS num_specializations
2 FROM dbo.athlete a
3 JOIN dbo.specialized sp ON a.athlete_code = sp.athlete_code
4 GROUP BY a.name
5 HAVING COUNT(sp.event_code) > 20
6 ORDER BY num_specializations DESC;
7
```

Data Output Messages Notifications

	name character varying (500)	num_specializations bigint
1	Ioannis Theofilakis	33
2	Alexandros Theofilakis	28
3	Gustaf Eric Carlberg	24
4	Frangiskos D. Mavrommatis	22
5	Gustaf Vilhelm Carlberg	22

10) Obtener la media de edad, junto con la media de la altura de los atletas redondeadas ambas a dos decimales, cuyo peso este informado, agrupado por el sexo.



The screenshot shows a PostgreSQL query editor interface. The top bar indicates the connection is 'SOURCE\_danidvo/STUDENT\_danidvo@PostgreSQL\_UOC'. Below the toolbar, the 'Query' tab is active, displaying the following SQL query:

```
1 SELECT sex,  
2     ROUND(AVG(age), 2) AS average_age,  
3     ROUND(AVG(height), 2) AS average_height  
4 FROM dbo.athlete  
5 WHERE weight IS NOT NULL  
6 GROUP BY sex;  
7
```

The 'Data Output' tab is also visible, showing the results of the query in a table format:

	sex character	average_age numeric	average_height numeric
1	M	26.40	179.52
2	F	25.01	168.98

#### 2.4.2. Corrección de sentencias SQL (5%)

Verificar que las siguientes consultas SQL están construidas y retornan los valores según los requerimientos definidos.

- 1) Se desea conocer la información de todos los atletas con 20 o 21 años de edad, cuyo nombre empiece por "Michael", y que la altura del atleta sea superior a 195 cm.

SOURCE\_danidvo/STUDENT\_danidvo@PostgreSQL\_UOC

Query Query History

```

1 SELECT name, *
2 FROM dbo.athlete
3 WHERE age = 20 or age = 21 and
4 name Like 'Michael%' and
5 height > 195;

```

Data Output Messages Notifications

	name character varying (500)	athlete_code integer	name character varying (500)	sex character	age integer	height integer	weight integer
1	Chung Chi Lok	18	Chung Chi Lok	M	20	179	70
2	Geertruida Christina (Gertrude-) "Truus" Baumeister	37	Geertruida Christina (Gertrude-) "Truus" Baumeister	F	20	[null]	[null]
3	Nicole Bobek (-Leal)	68	Nicole Bobek (-Leal)	F	20	165	55
4	Richard Walter Crooker	94	Richard Walter Crooker	M	20	198	93
5	Dorina Bczg	102	Dorina Bczg	F	20	160	48
6	Beaufort Burdekin	123	Beaufort Burdekin	M	20	[null]	[null]
7	Chen Bangping	156	Chen Bangping	F	20	177	69
8	Ahmed Bader Magour	159	Ahmed Bader Magour	M	20	190	90
9	Istvn Duds	181	Istvn Duds	M	20	175	95
10	Ralf Drecoll	212	Ralf Drecoll	M	20	185	82
11	Peter E. Devine	219	Peter E. Devine	M	20	188	82
12	Claire Chapotot	245	Claire Chapotot	F	20	169	63
13	Marinus Apolonia "Rinus" Bennaars	248	Marinus Apolonia "Rinus" Bennaars	M	20	[null]	[null]
14	Vladimir Andreyevich Chendarov	254	Vladimir Andreyevich Chendarov	M	20	176	66
15	Ricardo Capanema Esperard	284	Ricardo Capanema Esperard	M	20	[null]	[null]
16	Wolf-Hendrik Paul Beyer	288	Wolf-Hendrik Paul Beyer	M	20	200	82
17	Hans Eduard Btkofer-Perret	290	Hans Eduard Btkofer-Perret	M	20	[null]	[null]
18	Daniel Brunhart	293	Daniel Brunhart	M	20	168	60
19	DeVon D. Bean	354	DeVon D. Bean	M	20	175	70
20	Jos Mario Carrillo Zamudio	357	Jos Mario Carrillo Zamudio	M	20	174	69
21	Bernd Culmann	372	Bernd Culmann	M	20	180	71
22	Elta Cartwright (-Stromberg, -Henricksen)	384	Elta Cartwright (-Stromberg, -Henricksen)	F	20	165	48
23	Vasil Rozhilov	399	Vasil Rozhilov	M	20	[null]	[null]

Total rows: 1000 of 6430 Query complete 00:00:00.145 Ln 6, Col 1

**Problemas:**

1. Falta de paréntesis en las condiciones de edad
2. name y \* no deberían combinarse en SELECT, ya que \* ya incluye el campo name.

**Consulta corregida:** Para asegurar que las condiciones de edad se evalúen correctamente, se aplica el paréntesis.

SOURCE\_danidvo/STUDENT\_danidvo@PostgreSQL\_UOC

Query Query History

```

1 SELECT *
2 FROM dbo.athlete
3 WHERE (age = 20 OR age = 21)
4     AND name LIKE 'Michael%'
5     AND height > 195;

```

Data Output Messages Notifications

	athlete_code integer	name character varying (500)	sex character	age integer	height integer	weight integer
1	29541	Michael Allen "Mike" Bantom	M	20	203	93
2	54516	Michael Anthony Plumb, Jr.	M	21	196	95

- 2) Se desea conocer la información de todos los deportes cuyo nombre empiece por “At” y cuyos eventos deportivos contengan en la descripción los literales numéricos “100”, ordenados por la descripción del evento de forma ascendente.

Query

```

1 SELECT S.sport_description, E.event_description
2 FROM dbo.sport S, dbo.event E
3 WHERE S.sport_description LIKE 'At%' and
4 E.event_description LIKE '%100%'
5 ORDER BY E.event_description ASC;

```

Data Output

	sport_description character varying (100)	event_description character varying (500)
1	Athletics	Athletics Men's 100 metres
2	Athletics	Athletics Men's 4 x 100 metres Relay
3	Athletics	Athletics Women's 100 metres
4	Athletics	Athletics Women's 100 metres Hurdles
5	Athletics	Athletics Women's 4 x 100 metres Relay
6	Athletics	Cycling Men's 100 kilometres
7	Athletics	Cycling Men's 100 kilometres Team Time Trial
8	Athletics	Shooting Men's Small-Bore Rifle, 50 and 100 yards, Team
9	Athletics	Shooting Men's Small-Bore Rifle, Prone, 50 and 100 yar...
10	Athletics	Swimming Men's 100 metres Backstroke
11	Athletics	Swimming Men's 100 metres Breaststroke
12	Athletics	Swimming Men's 100 metres Butterfly
13	Athletics	Swimming Men's 100 metres Freestyle
14	Athletics	Swimming Men's 100 metres Freestyle For Sailors
15	Athletics	Swimming Men's 100 Yard Backstroke
16	Athletics	Swimming Men's 100 yard Freestyle
17	Athletics	Swimming Men's 4 x 100 metres Freestyle Relay
18	Athletics	Swimming Men's 4 x 100 metres Medley Relay
19	Athletics	Swimming Women's 100 metres Backstroke
20	Athletics	Swimming Women's 100 metres Breaststroke
21	Athletics	Swimming Women's 100 metres Butterfly
22	Athletics	Swimming Women's 100 metres Freestyle

Total rows: 24 of 24 Query complete 00:00:00.085 Ln 6, Col 1

### Problemas:

1. Falta de una relación explícita entre las tablas sport y event. Esto devolverá combinaciones no deseadas.
2. Es mejor usar JOIN para una mejor ejecución.

**Consulta corregida:** Aseguremos la relación entre sport y event mediante un JOIN en la clave sport\_code.

Query

```

3 JOIN dbo.event E ON S.sport_code = E.sport_code
4 WHERE S.sport_description LIKE 'At%'
5 AND E.event_description LIKE '%100%'
6 ORDER BY E.event_description ASC;
7

```

Data Output

	sport_description character varying (100)	event_description character varying (500)
1	Athletics	Athletics Men's 100 metres
2	Athletics	Athletics Men's 4 x 100 metres Relay
3	Athletics	Athletics Women's 100 metres
4	Athletics	Athletics Women's 100 metres Hurdles
5	Athletics	Athletics Women's 4 x 100 metres Relay



