NOTA: La estructura de los diferentes ejercicios es la siguiente: enunciado en negrita, declaración, pequeña explicación de la declaración que se ha empleado. En el caso, de los ejercicios complementarios indicados más abajo la estructura es similar pero se incluyen las tablas después del enunciado para una mejor comprensión.

En todas las declaraciones se utiliza la sentencia **SELECT** para consultar la información

**Enviar datos de todos los miembros**

SELECT \* FROM members;

Declaración que utiliza el asterisco para indicar que son todos los campos de la tabla members

**Enviar datos de todos los miembros**

SELECT \* FROM members;

Igual que anterior

**Se envió una lista de correo de un servicio ilegal en línea a la línea. Enviar correos electrónicos de participación y los detalles de las fechas de inscripción**

SELECT Email, Joined FROM maling\_list;

Declaración que utiliza los nombres de los campos que se quieren visualizar (Email y Joined) de la tabla mailing\_list

**Envíar todos los correos electronicos (Email), los nombres de pila (GivenName) y apellidos (FamilyName)**

SELECT Email, GivenName, FamilyName FROM mailing\_list;

Declaración que utiliza los nombres de los campos que se quieren visualizar (Email, GivenName y FamilyName) para indicar los campos de la tabla mailing\_list que se quieren visualizar

**Enviar datos de todos los apellidos (Surname) asegurando que no se repiten**

SELECT DISTINCT Surname FROM users;

Se utiliza Distinct para asegurar que sólo se muestran los valores del campo Surname diferentes, es decir, que ninguno se repita.

**Enviar datos de todos los usuarios ordenados en orden ascendente por su Email**

SELECT \* FROM users ORDER BY Email ASC;

Se utiliza la palabra clave ORDER BY para ordenar acompañada de ASC para indicar que tiene que ser de menor a mayor

**Enviar datos de todos los miembros ordenados por su HashedPassword en orden descendente**

SELECT \* FROM members ORDER BY HashedPassword DESC;

Igual que anterior pero con la palabra DESC para indicar de mayor a menor.

**Enviar datos de número de compra (NumberOfPurchases), hash de contraseñas(passwordHash), nombres de usuario (Username) ordenados por su hash de contraseñas(passwordHash) en orden descendente. Asegurarse que no haya duplicados**

SELECT DISTINCT NumberOfPurchase, PasswordHash, Username FROM subscribers ORDER BY PasswordHash DESC;

Igual que anterior pero se usa la palabra DISTINC para asegurar que no se repite.

**Enviar número de descargas, apellidos y tiempos de acceso ordenados por tiempos de acceso en orden ascendente y luego por número de descargas en orden descendente**

SELECT Download, LastName, AccesTime FROM users ORDER BY AccesTime ASC, Download DESC;

En este caso se ordena primero por campo AccesTime ne orden ascendente y en caso de que haya iguales se ordena en orden descendente por campo Download

**Enviar los datos del el top 3 de usuarios ordenados por apellidos en orden ascendente y número de publicaciones en orden descendente**

SELECT \* FROM users ORDER BY FamilyName, Post DESC LIMIT 3;

Se ordena en orden ascendente por le campo FamilyName y descendente, en caso de registros iguales en el campo FamilyName, por el campo Post. Se limita la consulta a los 3 primeros registros de la tabla (LIMIT 3)

**Enviar de los 3 miembros principales que se unieron fechas, nombres y nombre de usuario ordenados por fechas en orden descendente y luego por nombres en orden ascendente. Asegurarse que no hay repeticiones.**

SELECT DISTINCT JoinedOn, Name, Username FROM members ORDER BY JoinedOn DESC, Name ASC LIMIT 3;

En este caso al aplicar DISTINC a 3 campos (JoinedOn, Name, Username) se asegura que no hay repeticiones en las combinaciones posibles entre ellos pudiéndose dar repeticiones en los valores de cada campo individual. El resto de la declaración ya se ha explicado anteriormente.

**EJERCICIOS COMPLEMENTARIOS EXTRAIDOS DE KHAN ACADEMY**

Enlace: <https://es.khanacademy.org/computing/computer-programming/sql>

EJERCICIO 1

**Creamos una base de datos de canciones y artistas, y en este desafío vas a hacer listas de reproducción a partir de ellos**.

| [**artists**](javascript:void(0))12 rows |
| --- |
| id (PK)INTEGER |
| name TEXT |
| country TEXT |
| genre TEXT |

| [**songs**](javascript:void(0))11 rows |
| --- |
| id (PK)INTEGER |
| Artista TEXT |
| title TEXT |

**artists**

|  |  |  |  |
| --- | --- | --- | --- |
| id | name | country | genre |
| 1 | Taylor Swift | US | Pop |
| 2 | Led Zeppelin | US | Hard rock |
| 3 | ABBA | Sweden | Disco |
| 4 | Queen | UK | Rock |
| 5 | Celine Dion | Canada | Pop |
| 6 | Meatloaf | US | Hard rock |
| 7 | Garth Brooks | US | Country |
| 8 | Shania Twain | Canada | Country |
| 9 | Rihanna | US | Pop |
| 10 | Guns N' Roses | US | Hard rock |
| 11 | Gloria Estefan | US | Pop |
| 12 | Bob Marley | Jamaica | Reggae |

**songs**

| **id** | **artist** | **title** |
| --- | --- | --- |
| 1 | Taylor Swift | Shake it off |
| 2 | Rihanna | Stay |
| 3 | Celine Dion | My heart will go on |
| 4 | Celine Dion | A new day has come |
| 5 | Shania Twain | Party for two |
| 6 | Gloria Estefan | Conga |
| 7 | Led Zeppelin | Stairway to heaven |
| 8 | ABBA | Mamma mia |
| 9 | Queen | Bicycle Race |
| 10 | Queen | Bohemian Rhapsody |
| 11 | Guns N' Roses | Don't cry |

**En este primer paso, selecciona el title de todas las canciones cuyo artist sea 'Queen'.**

SELECT title FROM songs WHERE artist= 'Queen';

**Ahora vas a hacer una lista de reproducción 'Pop'. En preparación, selecciona el name (nombre) de todos los artistas del género 'Pop'.**

**(Ayuda: asegúrate de que lo escribas como 'Pop', SQL considera eso diferente de 'pop').**

SELECT name FROM artists WHERE genre = 'Pop';

**Para terminar de crear la lista de reproducción 'Pop', agrega otra consulta que seleccionará el title de todas las canciones de los artistas de 'Pop'. Debe usar IN en una subconsulta anidada que esté basada en tu consulta previa.**

SELECT title FROM songs WHERE artist IN (SELECT name FROM artists WHERE genre = 'Pop');

Es una declaración anidada donde a la declaración con SELECT se pone como filtro otra declaración con SELECT

EJERCICIO 2

**Creamos una base de datos de algunos autores populares y sus libros, con el conteo de palabras de cada libro. En este primer paso, selecciona todos los autores que han escrito más de 1 millón de palabras, usando GROUP BY y HAVING. Tu tabla de resultados debe incluir el 'author' y el conteo total de palabras como una columna 'total\_words'.**

|  |
| --- |
| [**books**](javascript:void(0))15 rows |
| id (PK)INTEGER |
| author TEXT |
| title TEXT |
| words INTEGER |

**books**

| **id** | **author** | **title** | **words** |
| --- | --- | --- | --- |
| 1 | J.K. Rowling | Harry Potter and the Philosopher's Stone | 79944 |
| 2 | J.K. Rowling | Harry Potter and the Chamber of Secrets | 85141 |
| 3 | J.K. Rowling | Harry Potter and the Prisoner of Azkaban | 107253 |
| 4 | J.K. Rowling | Harry Potter and the Goblet of Fire | 190637 |
| 5 | J.K. Rowling | Harry Potter and the Order of the Phoenix | 257045 |
| 6 | J.K. Rowling | Harry Potter and the Half-Blood Prince | 168923 |
| 7 | J.K. Rowling | Harry Potter and the Deathly Hallows | 197651 |
| 8 | Stephenie Meyer | Twilight | 118501 |
| 9 | Stephenie Meyer | New Moon | 132807 |
| 10 | Stephenie Meyer | Eclipse | 147930 |
| 11 | Stephenie Meyer | Breaking Dawn | 192196 |
| 12 | J.R.R. Tolkien | The Hobbit | 95022 |
| 13 | J.R.R. Tolkien | Fellowship of the Ring | 177227 |
| 14 | J.R.R. Tolkien | Two Towers | 143436 |
| 15 | J.R.R. Tolkien | Return of the King | 134462 |

SELECT author, sum (words) as total\_words FROM books GROUP BY author HAVING total\_words >1000000;

La declaración superior muestra los valores de los campos author y de un campo denominado total\_words que se genera de la suma de los valores del campo words, pero esta suma únicamente se realiza de los valores que correspondan a cada autor ya que lo hemos agrupado por autor (GROUP BY autor). Además, con la palabra HAVING establecemos que únicamente se visualicen los autores y cuyas palabras sumen más de 1000000 palabras

**Ahora selecciona todos los autores que escriben más de un promedio de 150,000 palabras por libro. Tu tabla de resultados debe incluir 'author' y el número promedio de palabras como una columna 'avg\_words'.**

SELECT author, AVG (words) as avg\_words FROM books GROUP BY author HAVING avg\_words >150000;

En este caso es lo mismo que el anterior pero en vez de sumar se haya la media de los valores de la columna words por autor.

EJERCICIO 3

**Creamos una base de datos para llevar el seguimiento de las calificaciones de los estudiantes, con su nombre, calificación numérica y qué porcentaje de actividades han completado. En este primer paso, selecciona todos los renglones y despliega name, number\_grade y percent\_completed, el cual puedes calcular multiplicando y redondeando la columna fraction\_completed.**

|  |
| --- |
| [**student\_grades**](javascript:void(0))6 rows |
| id (PK)INTEGER |
| name TEXT |
| number\_grade INTEGER |
| fraction\_completed REAL |

**student\_grades**

| **id** | **name** | **number\_grade** | **fraction\_completed** |
| --- | --- | --- | --- |
| 1 | Winston | 90 | 0.805 |
| 2 | Winnefer | 95 | 0.901 |
| 3 | Winsteen | 85 | 0.906 |
| 4 | Wincifer | 66 | 0.7054 |
| 5 | Winster | 76 | 0.5013 |
| 6 | Winstonia | 82 | 0.9045 |

SELECT name, number\_grade, ROUND (fraction\_completed\*100) as percent\_completed FROM student\_grades;

En esta declaración se muestran los valores de los campos name, number\_grade y el campo percent\_completed que se crea de la división entre 100 y redondeo posterior de los valores del campo fraction\_completed

**Ahora, este paso es un poco complicado. El objetivo es tener una tabla que muestre cuántos estudiantes han obtenido qué calificación letter\_grade. Puedes escribir letter\_grade usando CASE con la columna number\_grade, escribiendo 'A' para calificaciones > 90, 'B' para calificaciones > 80, 'C' para calificaciones > 70 y 'F' en cualquier otro caso. Después puedes usar COUNT con GROUP BY para mostrar el número de estudiantes con cada una de esas calificaciones.**

SELECT COUNT (\*) ,CASE

WHEN number\_grade > 90 THEN 'A'

WHEN number\_grade > 80 THEN 'B'

WHEN number\_grade > 70 THEN 'C'

ELSE 'F'

END AS letter\_grade

FROM student\_grades GROUP BY letter\_grade;

La sentencia CASE es como un condicional que establece unos valores, en este caso A,B,C y F en función de los valores de otro campo, en este caso, number\_grade. Finaliza con END y luego se crea la columna donde se quiere que se pongan los valores definidos anteriormente (lettergrade). En la declatación superior se cuenta el número de filas (COUNT(\*)) que hay de cada valor establecido en el CASE ya que lo hemos agrupado por los valores de la columna lettergrade.

EJERCICIO 4

**Creamos una tabla con todas las películas de 'Harry Potter', con una columna sequel\_id que coincide con el id de la secuela para cada película. Emite un SELECT que muestre el título de cada película junto al título de su secuela (o NULL si no tiene una secuela).**

|  |
| --- |
| [**movies**](javascript:void(0))8 rows |
| id (PK)INTEGER |
| titleTEXT |
| releasedINTEGER |
| sequel\_id |

**movies**

| **id** | **title** | **released** | **sequel\_id** |
| --- | --- | --- | --- |
| 1 | Harry Potter and the Philosopher's Stone | 2001 | 2 |
| 2 | Harry Potter and the Chamber of Secrets | 2002 | 3 |
| 3 | Harry Potter and the Prisoner of Azkaban | 2004 | 4 |
| 4 | Harry Potter and the Goblet of Fire | 2005 | 5 |
| 5 | Harry Potter and the Order of the Phoenix | 2007 | 6 |
| 6 | Harry Potter and the Half-Blood Prince | 2009 | 7 |
| 7 | Harry Potter and the Deathly Hallows – Part 1 | 2010 | 8 |
| 8 | Harry Potter and the Deathly Hallows – Part 2 | 2011 | NULL |

SELECT movies.title, movies\_sequel.title as sequel FROM movies LEFT OUTER JOIN movies as movies\_sequel ON movies.sequel\_id = movies\_sequel.id;

Esta declaración consiste en realizar un SELF JOIN, es decir, realizar una autounion de la tabla que tenemos con la propia tabla. Para ello se debe renombrar la tabla para que sea como si se realizara la unión con otra tabla ya que sino se genera un error por que no entiende que se quiera realizar una unión con la misma tabla.

EJERCICIO 5

**Creamos una base de datos para un sitio de redes de amigos, con una tabla que almacena datos de cada persona, una tabla con pasatiempos de cada persona y una tabla de conexiones de amistad entre las personas. En este primer paso, usa un JOIN para desplegar una tabla que muestre los nombres de las personas con sus pasatiempos.**

| [**persons**](javascript:void(0))5 rows |
| --- |
| id (PK)INTEGER |
| fullnameTEXT |
| ageINTEGER |

**persons**

| **id** | **fullname** | **age** |
| --- | --- | --- |
| 1 | Bobby McBobbyFace | 12 |
| 2 | Lucy BoBucie | 25 |
| 3 | Banana FoFanna | 14 |
| 4 | Shish Kabob | 20 |
| 5 | Fluffy Sparkles | 8 |

| [**hobbies**](javascript:void(0))10 rows |
| --- |
| id (PK)INTEGER |
| person\_idINTEGER |
| nameTEXT |

**hobbies**

| **id** | **person\_id** | **name** |
| --- | --- | --- |
| 1 | 1 | drawing |
| 2 | 1 | coding |
| 3 | 2 | dancing |
| 4 | 2 | coding |
| 5 | 3 | skating |
| 6 | 3 | rowing |
| 7 | 3 | drawing |
| 8 | 4 | coding |
| 9 | 4 | dilly-dallying |
| 10 | 4 | meowing |

| [**friends**](javascript:void(0))2 rows |
| --- |
| id (PK)INTEGER |
| person1\_idINTEGER |
| person2\_id |

**friends**

| **id** | **person1\_id** | **person2\_id** |
| --- | --- | --- |
| 1 | 1 | 4 |
| 2 | 2 | 3 |

SELECT persons.fullname, hobbies.name FROM persons JOIN hobbies ON persons.id = hobbies.person\_id;

Es una declaración de unión donde primero se definen los elementos que se quieren consultar (persons.fullname, hobbies.name) y luego se indica las tablas a la que pertenecen y por último la relación entre las tablas

**Ahora, usa otro SELECT con un JOIN para mostrar los nombres de cada par de amigos, con base en los datos en la tabla friends.**

SELECT persons.fullname as name1, amigos.fullname as name2 FROM persons JOIN friends ON friends.person1\_id = persons.id JOIN persons as amigos ON amigos.id= friends.person2\_id;

Igual que la anterior pero en este caso es una unión múltiple donde las tablas son 2 (persons y Friends) aunque hay que establecer una relación entre un campo de la primera tabla ( persons.id)con otro de la segunda (friends.person1\_id) y luego una segunda relación de un segundo campo de esta segunda tabla (friends.person2\_id) con el mismo campo de la primera con el que ya se había establecido una primera relación. Es por ello que hay que renombrar a la primera tabla para realizar esta segunda relación (persons as amigos)