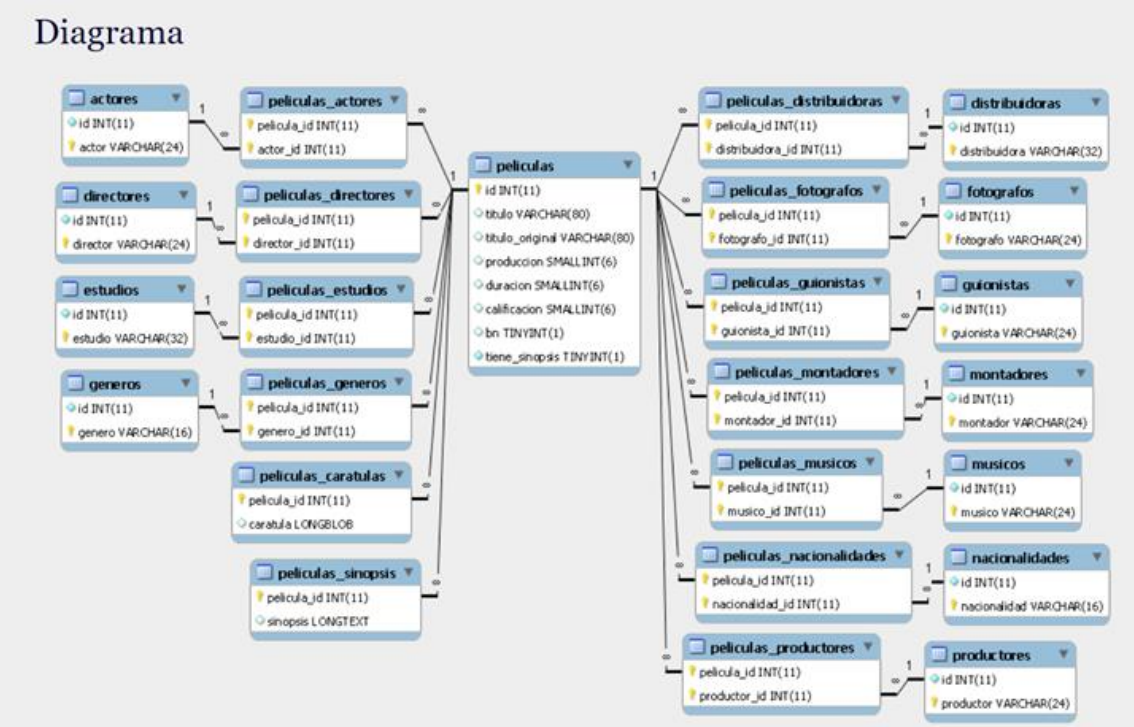
Boletín de ejercicios Tema 3

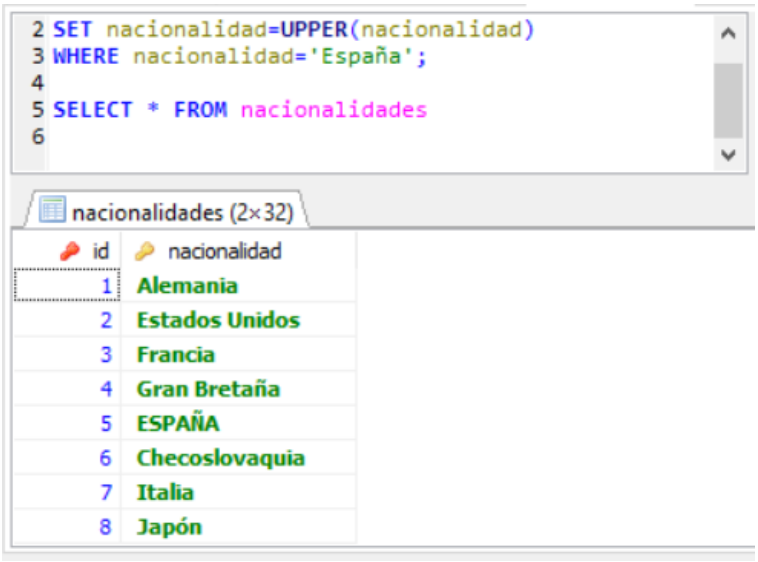
NOTA: La resolución de cada uno de los ejercicios debe ser una captura de pantalla en la que se refleje la consulta y los resultados obtenidos de ella. Para los cuatro primeros ejercicios se debe utilizar la interfaz HeideSQL de MaríaDB y la base de datos filmoteca, que os he subido junto al boletín. El diagrama de dicha base es:



Ejercicio 1

Actualiza la tabla **nacionalidades** de modo que España aparezca en mayúsculas. Muestra la sentencia SQL y el resultado.

Solución



```
2 SET nacionalidad=UPPER(nacionalidad)
3 WHERE nacionalidad='España';
4
5 SELECT * FROM nacionalidades
6
```

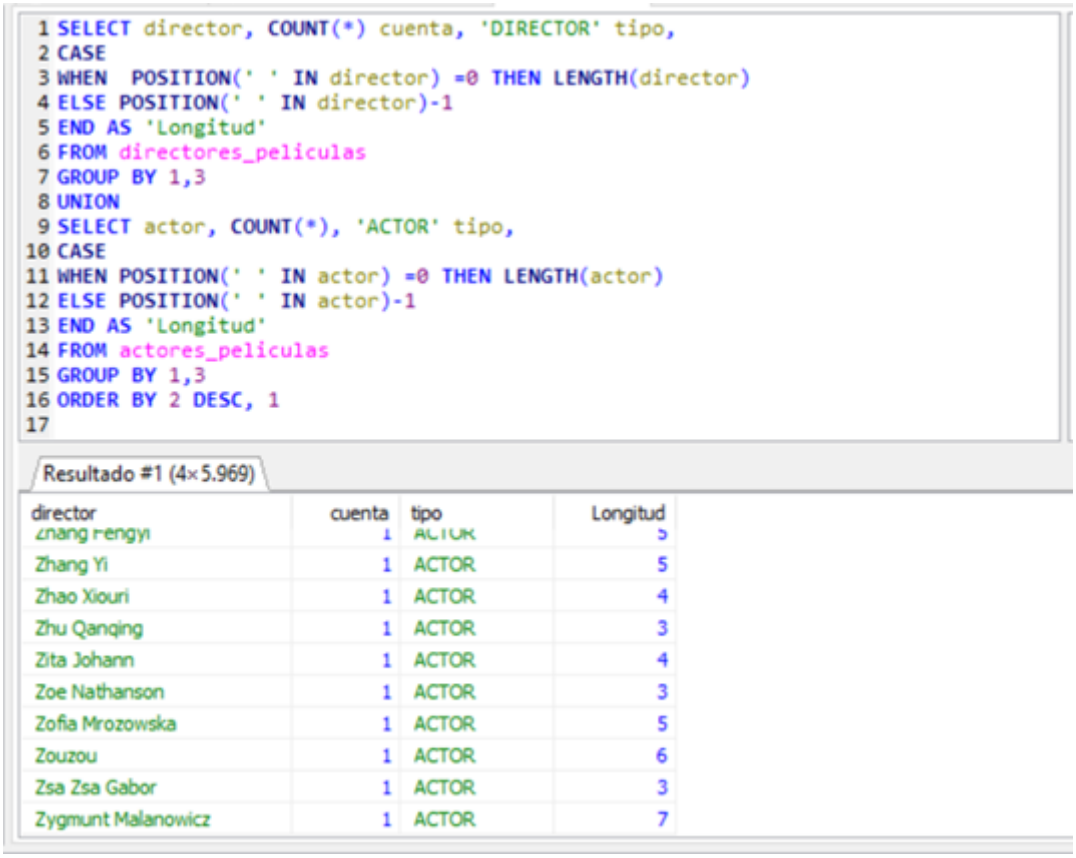
nacionalidades (2×32)

id	nacionalidad
1	Alemania
2	Estados Unidos
3	Francia
4	Gran Bretaña
5	ESPAÑA
6	Checoslovaquia
7	Italia
8	Japón

Ejercicio 2

Obtener un listado de los directores y actores ordenándolos por orden descendente del número de películas en las que han participado. En el listado debe figurar en una columna el nombre del actor o director, en otra el número de películas, en otra si es director o autor y en la última el número de letras que contienen sus nombres (sin apellidos).

Solución



```
 1 SELECT director, COUNT(*) cuenta, 'DIRECTOR' tipo,
 2 CASE
 3 WHEN  POSITION(' ' IN director) =0 THEN LENGTH(director)
 4 ELSE POSITION(' ' IN director)-1
 5 END AS 'Longitud'
 6 FROM directores_peliculas
 7 GROUP BY 1,3
 8 UNION
 9 SELECT actor, COUNT(*), 'ACTOR' tipo,
10 CASE
11 WHEN POSITION(' ' IN actor) =0 THEN LENGTH(actor)
12 ELSE POSITION(' ' IN actor)-1
13 END AS 'Longitud'
14 FROM actores_peliculas
15 GROUP BY 1,3
16 ORDER BY 2 DESC, 1
17
```

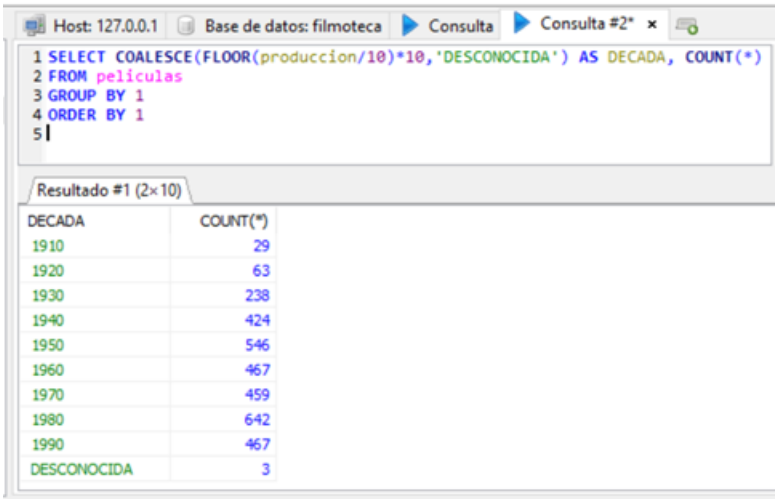
Resultado #1 (4×5.969)

director	cuenta	tipo	Longitud
Zhang Yi	1	ACTOR	5
Zhao Xiouri	1	ACTOR	4
Zhu Qanqing	1	ACTOR	3
Zita Johann	1	ACTOR	4
Zoe Nathanson	1	ACTOR	3
Zofia Mrozowska	1	ACTOR	5
Zouzou	1	ACTOR	6
Zsa Zsa Gabor	1	ACTOR	3
Zygmunt Malanowicz	1	ACTOR	7

Ejercicio 3

Teniendo en cuenta que para dividir se utiliza “/”, para multiplicar “*” y recordando funciones de redondeo que hemos visto, averiguar, según los datos que tenemos en nuestra base de datos de cine, cuántas películas se hicieron en cada década mostrando número de películas y década. Si hay películas sin fecha en su lugar debe poner ‘Desconocida’.

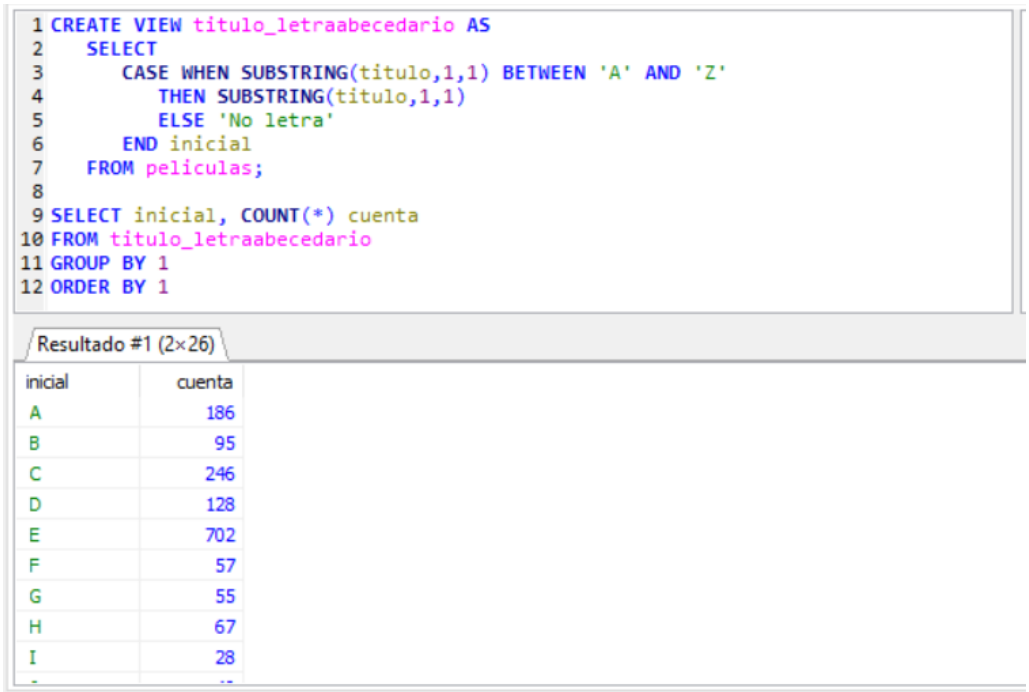
Solución:



Ejercicio 4

Mostrar en dos columnas, **Letra** y **Nº de películas**, el número de películas que comienzan por cada letra del abecedario. Las películas que no empiecen por una letra se deben agrupar en un grupo "**No letra**".

Solución:



Ejercicio 5

Crea la base de datos de la siguiente manera: **CREATE DATABASE colegio;**

Para que en las sucesivas consultas utilicemos esta base de datos tecleamos: **USE colegio;**

Crea la tabla, denominada **estudiantes**, con los siguientes campos:

- **Id** (valor numérico)
- **nombre** (secuencia de, como máximo, 50 caracteres). Esta columna no puede tomar valores nulos
- **nif** (secuencia de 9 caracteres)
- **nacimiento** (es una fecha)
- **observaciones** (secuencia de, como máximo, 100 caracteres)

Al final de la tabla se debe especificar:

- Con el nombre **pk_alumno**, la restricción que especifica que una de las variables anteriores es clave primaria de la tabla.
- Con el nombre **uq_nif** que los valores de la columna nif no pueden repetirse

Inserta los siguientes valores en la tabla acabas de crear especificando todas las columnas en las que se van a insertar:

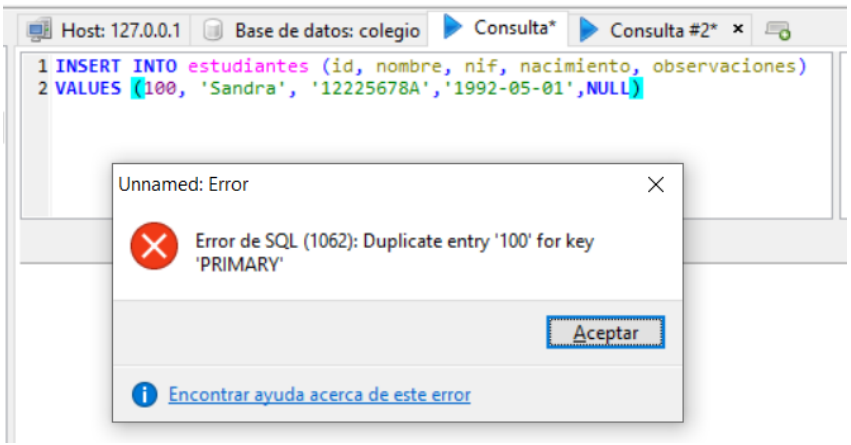
- (100, 'Beatriz', '12345678A','1992-01-01', NULL)
- (101, 'Yolanda', '23456789B','1972-02-02', NULL)
- (102, 'Ebba', '34567890C','1980-06-08', NULL)
- (103, 'Sonia', '45678901D','1982-01-22', NULL)
- (104, 'Pedro', '56789012E','1992-01-01', NULL)

Genera un error al insertar un nuevo valor debido a la restricción **pk_alumno** y realiza una captura de pantalla.

Solución:

```
CREATE DATABASE colegio;

USE colegio;

CREATE TABLE estudiantes (
   id              INTEGER,
   nombre          VARCHAR(50) NOT NULL,
   nif             CHAR(9),
   nacimiento      DATE,
   observaciones   VARCHAR(100),

   CONSTRAINT pk_estudiante PRIMARY KEY (id),
   CONSTRAINT uq_nif UNIQUE KEY (nif)
);
```

```
1 INSERT INTO estudiantes (id, nombre, nif, nacimiento, observaciones)
2 VALUES (100, 'Beatriz', '12345678A','1992-01-01', NULL),
3        (101, 'Yolanda', '23456789B','1972-02-02', NULL),
4        (102, 'Ebba', '34567890C','1980-06-08', NULL),
5        (103, 'Sonia', '45678901D','1982-01-22', NULL),
6        (104, 'Pedro', '56789012E','1992-01-01', NULL)
7
```



Ejercicio 6

Crea la tabla, denominada **docentes**, con los siguientes campos:

- Id (valor numérico)
- nombre (secuencia de, como máximo, 50 caracteres). Esta columna no puede tomar valores nulos

- nif (secuencia de 9 caracteres)
- nacimiento (es una fecha)
- observaciones (secuencia de, como máximo, 100 caracteres)

Al final de la tabla se debe especificar:

- Con el nombre pk_docente, la restricción que especifica que una de las variables anteriores es clave primaria de la tabla.
- Con el nombre uq_nif que los valores de la columna nif no pueden repetirse

Inserta los siguientes valores en la tabla que acabas de crear sin especificar las columnas en las que se van a insertar:

- (10, 'Andrea', '25874469A','1968-05-01', NULL)
- (11, 'Gerardo', '78552369B','1960-11-02', NULL)
- (12, 'María', '36046789C','1965-02-25', NULL)
- (13, 'Janet', '36124418D','1970-03-22', NULL)
- (14, 'Javier', '52876631F','1970-06-11', NULL)

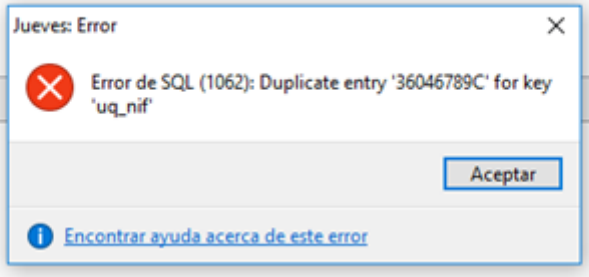
Genera un error en los campos nombre y nif al insertar filas. Realiza capturas de pantalla de los mismos.

Solución:

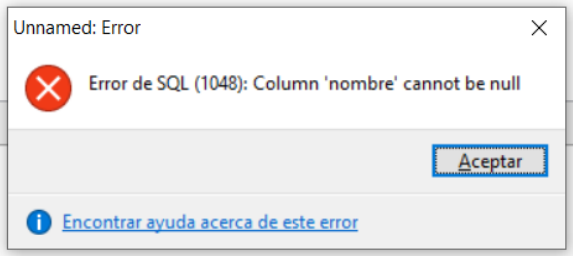
```sql
USE colegio;

CREATE TABLE docentes (
    id                INTEGER,
    nombre            VARCHAR(50) NOT NULL,
    nif               CHAR(9),
    nacimiento        DATE,
    observaciones     VARCHAR(100),

    CONSTRAINT pk_docentes PRIMARY KEY (id),
    CONSTRAINT uq_nif UNIQUE KEY (nif)
);
```

```sql
INSERT INTO docentes
VALUES (10, 'Andrea', '25874469A','1968-05-01', NULL),
       (11, 'Gerardo', '78552369B','1960-11-02', NULL),
       (12, 'María', '36046789C','1965-02-25', NULL),
       (13, 'Janet', '36124418D','1970-03-22', NULL),
       (14, 'Javier', '52876631F','1970-06-11', NULL)
```

```sql
INSERT INTO docentes (id,nombre,nif,nacimiento,observaciones)
VALUES (25,'Jessica','36046789C','1972-11-22',NULL)
```




```sql
INSERT INTO docentes (id,nombre,nif,nacimiento,observaciones)
VALUES (13, NULL, '12333678A','1972-05-01',NULL)
```



Ejercicio 7

Crea una tabla, denominada **aulas**, con los siguientes campos:

- **id** (valor numérico)
- **ubicación** (secuencia de tres caracteres)
- **capacidad** (valor numérico)

Para que todo sea correcto, añade las restricciones que consideres y en donde consideres.

Inserta en dicha tabla los siguientes valores:

- (1001,'A01',15)
- (1002,'A02',15)
- (1014,'A14',16)

Solución:

```
1 CREATE TABLE aulas (
2    id          INTEGER,
3    ubicación   CHAR(3),
4    capacidad   INTEGER,
5
6    CONSTRAINT pk_aula PRIMARY KEY (id),
7    CONSTRAINT uq_ubicación UNIQUE KEY (ubicación));
```

```
1 INSERT INTO aulas
2 VALUES (1001,'A01',15),(1002,'A02',15),(1014,'A14',16)
3
```

Ejercicio 8

Crear las siguientes tablas teniendo en cuenta que dependen de otras (no considerar en este ejercicio modificación o eliminación de datos). Añade las restricciones que consideres y donde consideres:

- Tabla, denominada **cursos**, con los siguientes campos:
    - **id** (valor numérico)
    - **docente_id** (valor numérico)
    - **aula_id** (valor númerico)
    - **nombre** (secuencia de, como máximo 50 caracteres)
    - **fecha_inicio** (almacena una fecha)
    - **fecha_fin** (almacena una fecha)
    - **horario** (secuencia de, como máximo 50 caracteres)

- Tabla, denominada **clases**, con los siguientes campos:
    - **id** (valor numérico)
    - **curso_id** (valor numérico)
    - **fecha** (almacena una fecha)
    - **contenido** (secuencia de caracteres de, como máximo, 100 caracteres)

- Tabla **estudiantes_cursos**
    - **estudiante_id** (valor numérico)
    - **curso_id** (valor numérico)

Solución:

```
CREATE TABLE cursos (
   id              INTEGER,
   docente_id      INTEGER,
   aula_id         INTEGER,
   nombre          INTEGER,
   fecha_inicio    DATE,
   fecha_fin       DATE,
   horario         VARCHAR(50),

   CONSTRAINT pk_curso PRIMARY KEY (id),
   CONSTRAINT fk_docente FOREIGN KEY (docente_id)
         REFERENCES docentes(id),
   CONSTRAINT fk_aula FOREIGN KEY (aula_id)
         REFERENCES aulas(id)
);

```

```
CREATE TABLE clases (
   id              INTEGER,
   curso_id        INTEGER,
   fecha           DATE,
   contenido       VARCHAR(100),

   CONSTRAINT pk_clase PRIMARY KEY (id),
   CONSTRAINT uq_clase UNIQUE KEY (curso_id, fecha),
   CONSTRAINT fk_curso FOREIGN KEY (curso_id)
         REFERENCES cursos(id)
);

```

```
CREATE TABLE estudiantes_cursos (
   estudiante_id     INTEGER,
   curso_id          INTEGER,

   CONSTRAINT pk_estudiantecurso PRIMARY KEY (estudiante_id, curso_i
   CONSTRAINT fk_estudiante FOREIGN KEY (estudiante_id)
      REFERENCES estudiantes(id),
   CONSTRAINT fk_curso2 FOREIGN KEY (curso_id)
      REFERENCES cursos(id)
);
```

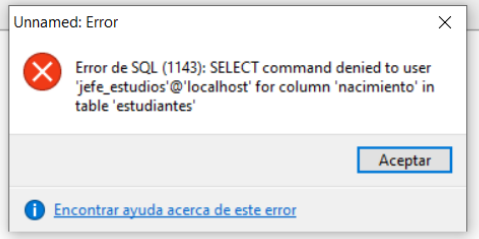
Ejercicio 9

Utilizando el icono de usuarios de HeidiSQL, crea un usuario denominado **jefe_estudios** de modo que pueda acceder a la tabla **estudiantes** pero sólo puede consultar el nombre y el NIF del estudiante. Configura este usuario de modo que puede delegar este permiso en otro usuario. Demuestra que no puede acceder a otra tabla o a otra columna de la tabla estudiantes.

Solución:

```
1 GRANT SELECT (nif,nombre)
2 ON estudiantes
3 TO jefe_estudios@localhost
4 WITH GRANT OPTION
5
```

```
1 SELECT nacimiento
2 FROM estudiantes
```




```
1 SELECT *
2 FROM docentes
```