# Práctica 3. MANIPULACIÓN DE DATOS EN SQL

## 0. Introducción

Esta práctica se dedica al estudio de las sentencias SQL de manipulación de datos (DML) que permiten consultar, modificar, insertar y borrar la información almacenada en una tabla.

En esta práctica las sentencias de manipulación que se estudiarán serán las más básicas, dejando para prácticas posteriores las formas más complejas.

Los ejemplos utilizados en esta práctica se refieren a la tabla de ALUMNOS. El contenido de dicha tabla se muestra a continuación.

**ALUMNOS**

| NUM_EXP | TITULACION | FECHA_APERTURA_EXP | NOTA_MEDIA | FECHA_NACIMIENTO | NIF | |
|---|---|---|---|---|---|---|
| 1 | 101 | 30/09/02 | | 12/02/81 | 07345128H | ... |
| 2 | 101 | 15/10/01 | | 16/12/79 | 27365138A | ... |
| 3 | 102 | 15/10/01 | | 06/08/80 | 10134562B | ... |
| 4 | 102 | 12/10/99 | 6,75 | 20/07/75 | 57176355K | ... |
| 5 | 103 | 13/10/97 | 7,05 | 13/02/76 | 08124356C | ... |

| | APELLIDO1 | APELLIDO2 | NOMBRE | DOMICILIO | COD_POSTAL | POBLACION | COD_PROV |
|---|---|---|---|---|---|---|---|
| .... | PEREZ | PEREZ | JUAN | CL/ PARIS, 20 | 10005 | CACERES | 10 |
| ... | GARCIA | SANCHEZ | PAULA | CL/ PARRAS, 6 | 10001 | CACERES | 10 |
| ... | LARRA | ALVAREZ | PEDRO | PL/ MAYOR, 15 | 10001 | CACERES | 10 |
| ... | VAZQUEZ | PEDROSA | IRENE | PL/ MAYOR, 4 | 10001 | CACERES | 10 |
| ... | TENA | TENA | ALBERTO | CL/ ITALIA, 8 | 6001 | BADAJOZ | 6 |

Esta tabla ha sido creada y pertenece al esquema PRACTBD y puede ser accedida desde cualquier usuario con acceso a la BD.

# 1. Consultas simples en SQL

## *1.1 Sintaxis de la sentencia SELECT*

Todas las consultas sobre tablas en una base de datos se llevan a cabo con la sentencia de SQL SELECT. Su sintaxis es la siguiente:

```
SELECT  [ALL | DISTINCT]  atributos
      FROM  [esquema.]tablas
      [WHERE  condición [Subconsulta] ]                    (1)
      [GROUP BY atributos [ HAVING predicado de grupo] ]
{[UNION | UNION ALL | INTERSECT | MINUS] selección }       (2)
      [ORDER BY atributos  [ASC | DESC] ] ;                (3)
```

(1) condición [Subconsulta].- Quiere indicar que en la condición de selección pueden aparecer diferentes consultas.
(2) selección.- Hace referencia a otra sentencia **Select**.
(3) ASC significa en orden ascendente, DESC en orden descendente.

En la cláusula WHERE se pueden utilizar expresiones en las que intervengan atributos y valores constantes para realizar la selección de las correspondientes tuplas. Los predicados más usuales que pueden construirse utilizan los operadores y funciones que se definen el apartado 2.

## *1.2 Selección*

En Álgebra relacional la operación de selección se representa como $\sigma_{\text{condicion}}$**(nombre de tabla)**.

En el lenguaje SQL la operación de selección se realiza mediante la sentencia **SELECT.**

Inicialmente utilizaremos el * para indicar que deseamos consultar todos los atributos de una tabla.

**Ejemplo:** Seleccionar toda la información de la tabla de alumnos:

```
SELECT *
FROM practbd.alumnos;
```

```
NUM_EXP    TITULACION FECHA_AP NOTA_MEDIA FECHA_NA NIF       ...
---------- ---------- -------- ---------- -------- --------- ---
         1        101 30/09/02            12/02/81 07345128H ...
         2        101 15/10/01            16/12/79 27365138A ...
         3        102 15/10/01            06/08/80 10134562B ...
         4        102 12/10/99       6,75 20/07/75 57176355K ...
         5        103 13/10/97       7,05 13/02/76 08124356C ...
```

Como la tabla que estamos consultando pertenece a un esquema distinto que el de conexión, debemos indicar el nombre del esquema propietario de la tabla (practbd)

Las condiciones que han de cumplir las tuplas para que sean seleccionadas se ponen dentro de la cláusula **WHERE.**

**Ejemplo:** Seleccionar toda la información de la tabla de alumnos del expediente 1:

```
/*    σ ( num_exp = 1)(alumnos)  */

SELECT *
FROM practbd.alumnos
WHERE num_exp = 1;
```

```
NUM_EXP    TITULACION FECHA_AP NOTA_MEDIA FECHA_NA NIF       ...
---------- ---------- -------- ---------- -------- --------- ---
         1        101 30/09/02            12/02/81 07345128H ...
```

### 1.2.1 Consulta mediante patrones

% Cualquier número de caracteres
_ Un sólo carácter.

**Ejemplo:** Seleccionar toda la información de los alumnos cuya población tenga una 'E' como penúltima letra.

/* $\sigma_{(\text{poblacion LIKE '\%E\_'})}(\text{alumnos})$ */

```
SELECT *
FROM practbd.alumnos
WHERE poblacion LIKE '%E_';

NUM_EXP    TITULACION FECHA_AP NOTA_MEDIA FECHA_NA POBLACION ...
---------- ---------- -------- ---------- -------- --------- ---
         1        101 30/09/02            12/02/81 CACERES   ...
         2        101 15/10/01            16/12/79 CACERES   ...
         3        102 15/10/01            06/08/80 CACERES   ...
         4        102 12/10/99       6,75 20/07/75 CACERES   ...
```

## *1.3 Proyección*

Cuando no se seleccionan todos los atributos se deben incluir los atributos seleccionados en la cláusula SELECT. En Álgebra relacional la operación de proyección se representa como $\Pi_{\text{listacolumnas}}$ **(tabla)**.

**Ejemplo:** Realizar la proyección sobre el atributo *poblacion* en la tabla de *alumnos*:

```
SELECT poblacion
FROM practbd.alumnos;

POBLACION
------------------------
CACERES
CACERES
CACERES
CACERES
BADAJOZ
```

Se puede cambiar en el resultado la cabecera de la columna utilizando la palabra reservada AS. Ejemplo:

```
SELECT población AS municipio
FROM practbd.alumnos;

MUNICIPIO
------------------------
CACERES
CACERES
CACERES
CACERES
BADAJOZ
```

Oracle permite sustituir la palabra reservada AS por un espacio en blanco para realizar la misma función.

### 1.3.1 Supresión de valores repetidos en la consulta

Para que no aparezcan valores duplicados en el resultado de la selección añadimos la cláusula DISTINCT después del SELECT.

**Ejemplo:** Igual que la anterior pero eliminando los valores duplicados:

```
/*    Π Poblacion (alumnos) */

SELECT DISTINCT poblacion
FROM practbd.alumnos;

POBLACION
------------------------
BADAJOZ
CACERES
```

### 1.3.2 Ordenación de la salida

Se puede pedir que la consulta salga ordenada por alguno de los atributos seleccionados en el SELECT mediante ORDER BY. Se puede ordenar por más de un atributo, separando el nombre de estos por comas y en el caso de que se desee ordenar un atributo de mayor a menor valor, añadiremos al nombre del atributo la cláusula DESC.

**Ejemplo:** Seleccionar dni, apellidos, nombre y población de todos los alumnos ordenando descendentemente por la población y ascendentemente por el primer apellido

```
SELECT dni, apellido1,apellido2,nombre, poblacion
FROM  practbd.alumnos
ORDER BY poblacion DESC, apellido1;

NIF       APELLIDO1     APELLIDO2     NOMBRE          POBLACION
--------- ------------- ------------- --------------- ---------------
27365138A GARCIA        SANCHEZ       PAULA           CACERES
10134562B LARRA         ALVAREZ       JUAN PEDRO      CACERES
07345128H PEREZ         PEREZ         JUAN            CACERES
57176355K VAZQUEZ       PEDROSA       IRENE           CACERES
08124356C TENA          TENA          ALBERTO         BADAJOZ
```

## *1.4 Creación de tablas basadas en una consulta*

Además de la forma vista en la práctica anterior para la creación de tablas, podemos crear una tabla basándonos en el resultado de una consulta. En este caso además de la creación se dan de alta dentro de la tabla todos los tuplas seleccionadas.

**Sintaxis :**

```
CREATE TABLE nombre_tabla
AS consulta;
```

**Ejemplo:** Crear una tabla que contenga toda la información de los alumnos de la titulacion 101.

```
CREATE TABLE alumnos101
AS SELECT *
   FROM practbd.alumnos
   WHERE titulacion=101;
```

# 2. Operadores y funciones

## *2.1 Operadores*

En la formación de las condiciones se pueden utilizar los siguientes tipos de operadores:

### 2.1.1 Operadores de relación

| | |
|---|---|
| **=** | Igual |
| **>** | Mayor |
| **<** | Menor |
| **> =** | Mayor o igual |
| **< =** | Menor o igual |
| **!=** o **< >** | Distinto |
| **IS NULL** | Es nulo |
| **IS NOT NULL** | Es no nulo |
| **BETWEEN valor1 AND valor2** | (Valores comprendidos entre valor1 y valor2) |

### 2.1.2 Operadores lógicos

| | |
|---|---|
| **AND** | y |
| **OR** | o |
| **NOT** | No |

## *2.2 Funciones aplicables a una sola tupla*

Se aplican por cada una de las tuplas de la tabla consultada. Estas funciones pueden aparecer en la cláusula SELECT aplicadas a uno o varios de los atributos seleccionados, en la cláusula WHERE y en la cláusula HAVING para consultas agrupadas (se verán más adelante).

### 2.2.1 Funciones numéricas

Aceptan y devuelven valores numéricos.

| Nombre | Descripción |
|---|---|
| ABS(n) | Devuelve el valor absoluto de n |
| CEIL(n) | Devuelve el entero mayor o igual a n |
| FLOOR(n) | Devuelve el entero menor o igual a n |
| MOD(m,n) | Devuelve el resto de la división de m por n |
| POWER (m,n) | Devuelve m elevado a n |
| ROUND (n,[m]) | Devuelve n redondeado con m decimales |
| SIGN (n) | Si n<0 devuelve -1, si n=0 devuelve 0, y si n>0 devuelve 1 |
| TRUNC (n[,m]) | Devuelve n truncado a m plazas decimales. Si se omite m será 0. Si m es negativo trunca m dígitos a la izquierda del punto decimal. |

### 2.2.2 Funciones de conversión de tipos

Convierten valores de un tipo de datos a otro.

| Nombre | Descripción |
|---|---|
| TO_CHAR (n) | Convierte un número o fecha en cadena de caracteres (VARCHAR2) |
| TO_DATE (c) | Convierte una cadena de caracteres a una fecha (DATE) |
| TO_NUMBER (c) | Convierte una cadena de caracteres a un número (NUMBER). |
| CAST (expr AS tipo) | Se utiliza para tomar un valor o expresión de un tipo y devolver un valor de otro tipo |

El parámetro tipo de las funciones CAST puede tener uno de los siguientes valores:

- BINARY[(N)]
- CHAR[(N)]
- DATE
- DATETIME
- DECIMAL[(M[,D])]
- SIGNED [INTEGER]
- TIME
- UNSIGNED [INTEGER]

### 2.2.3 Funciones de cadenas

Aceptan y devuelven cadenas de caracteres, excepto INSTR Y LENGTH que devuelven valores numéricos.

| Nombre | Descripción |
|---|---|
| RPAD (c1,n,c2) | Devuelve **c1** relleno por la derecha con tantas ocurrencias de **c2** como sean necesarias hasta que la longitud de **c1** sea **n**. |
| LPAD (c1,n,c2) | Devuelve **c1** relleno por la izquierda con tantas ocurrencias de **c2** como sean necesarias hasta que la longitud de **c1** sea **n**. |
| RTRIM (c1, c2) | Devuelve **c1**, borrando por la derecha de **c1** todos los caracteres que aparecen en **c2**. |
| LTRIM (c1, c2) | Devuelve **c1**, borrando por la izquierda todos los caracteres que aparecen en **c2**. |
| TRIM (c1,c2) | Devuelve **c1**, borrando por la derecha y la izquierda de **c1** todos los caracteres que aparecen en **c2**. Si no se especifica **c2** se toma como espacio en blanco. |
| LOWER (c) | Pasa **c** a minúsculas |
| UPPER(c) | Pasa **c** a mayúsculas |
| INITCAP(c) | Pone en mayúsculas la primera letra de cada palabra de **c** |
| LENGTH(c) | Nos devuelve el numero de caracteres de **c** |
| SUBSTR(c, n1, [n2]) | Nos devuelve **n2** carácteres comenzando desde la posición **n1** de **c**. |
| INSTR (c,c2, n1, n) | Nos devuelve la posición de la **n**-ésima ocurrencia de patrón **c2** en la cadena **c** comenzando la búsqueda en la posición **n1** de **c** |
| REPLACE (c,c1,c2) | Nos devuelve la cadena **c**, sustituyendo **c1** por **c2** |

### *2.3 Funciones de grupos*

Devuelven resultados basados en grupos de tuplas. Pueden aparecer en las cláusulas SELECT, HAVING Y ORDER BY. **Nunca pueden aparecer en la cláusula WHERE**. Las que más utilizaremos son:

| Nombre | Descripción |
|---|---|
| AVG(c) | Devuelve la media de los valores de **c** |
| COUNT(a) | Devuelve el número de valores de ***a*** no nulos |
| COUNT(*) | Devuelve el número de tuplas recuperadas. |
| COUNT(DISTINCT a) | Devuelve el número de valores distintos no nulos de **a** |
| COUNT(DISTINCT *) | Devuelve el número de tuplas distintas recuperadas |
| MAX(c) | Devuelve el valor máximo de **c** en el grupo |
| MIN (c) | Devuelve el valor mínimo de c en el grupo |
| SUM(c) | Devuelve la suma de los valores de c en el grupo |
| VAR(c) | Devuelve la varianza de c |
| STD(c) | Devuelve la desviación estándar de los valores de c en el grupo |

Donde: “c” puede ser un atributo o una lista de valores y “a” un atributo o *.

## 3. Secuencias

Una secuencia es un objeto de la base de datos en Oracle, desde el que un usuario puede generar enteros únicos. Se puede utilizar la secuencia para generar automáticamente valores de clave única.

**Sintaxis :**

```
CREATE SEQUENCE nombre_secuencia
            [INCREMENT BY entero]   [START WITH entero]
            [MAXVALUE entero] [MINVALUE entero]
            [CYCLE | NOCYCLE] ;
```

Para utilizar una secuencia se utilizan las funciones:

- **nombre_secuencia.NEXTVAL:** Devuelve el valor de la secuencia y la incrementa
- **nombre_secuencia.CURRVAL:** Devuelve el valor actual de la secuencia.

Para eliminar una secuencia se utiliza la sentencia:

```
DROP SEQUENCE nombre_secuencia;
```

**Ejemplo:** Crear una secuencia para los números de expediente de los alumnos cuyo primer valor sea 6 y el último valor el 999999 y se incremente de 1 en 1

```
CREATE SEQUENCE snum_exp
         INCREMENT BY 1 START WITH 6 MAXVALUE 999999;
```

# 4. Inserción de tuplas

## *4.1 Inserción de una única tupla*

**Sintaxis :**

```
INSERT INTO [esquema.]tabla [(atributo1,...,atributon)]
VALUES (valor_atributo1,...,valor_atributon);
```

Si se ponen los valores de todos los atributos en el mismo orden en el que se definieron los atributos en la tabla, no es necesario poner el nombre de los atributos. Es recomendable indicar los atributos para que no fallen las sentencias INSERT incluidas en programas, después de haber añadido nuevos atributos a la tabla.

**Ejemplo:** Dar de alta un nuevo alumno en **alumnos101** cuyas características son las siguientes:

- Nombre y apellidos: Ruben Osado Martinez
- NIF: 79456874F
- Fecha de matriculación: La fecha en el momento del alta
- Fecha de nacimiento: 28 de Mayo de 1985
- Dirección: C\ Lima, 25 10005 Caceres, código de provincia 10

```
INSERT INTO alumnos101 (nombre, apellido1, apellido2, nif, num_exp,
                        titulacion, fecha_nacimiento, domicilio,
                        cod_postal, poblacion, cod_prov)
VALUES ('RUBEN', 'OSADO', 'MARTINEZ', '79456874F', snum_exp.NEXTVAL,
        101, '28/05/85', 'CL/ LIMA, 25', 10005, 'CACERES', 10);


1 fila creada.
```

## *4.2 Inserción de tuplas basadas en el resultado de una consulta*

**Sintaxis:**

```
INSERT INTO [esquema.]tabla [(atributo1,...,atributon)]
Consulta;
```

**Ejemplo:** Dar de alta al alumno con nif 10134562B en la titulación 101.

```
INSERT INTO alumnos101 (nombre, apellido1, apellido2, nif, num_exp,
                        titulacion, fecha_nacimiento, domicilio,
                        cod_postal, poblacion, cod_prov)
SELECT nombre, apellido1, apellido2,nif, snum_exp.NEXTVAL,101,
       fecha_nacimiento, domicilio,cod_postal,poblacion,cod_prov
FROM practbd.alumnos
WHERE nif='10134562B';

1 fila creada.
```

### *4.3 Inserción de múltiples tuplas*

**Sintaxis:**

```
INSERT INTO [esquema.]tabla [(atributo1,...,atributon)]
VALUES (valor_atributo1,...,valor_atributon),
       (valor_atributo1,...,valor_atributon),... ;
```

**IMPORTANTE**: Esta opción, que está implementada en varios SGBD como MySQL, PostgreSQL o DB2 **NO puede usarse en Oracle.** Oracle permite hacer algo similar pero usando el lenguaje PL/SQL.

## 5. Actualización de tuplas

**Sintaxis :**

UPDATE [esquema.]tabla

SET atributo1=valor [,atributo2=valor2 ...]

[WHERE expresión]

**Si a una sentencia UPDATE no se le incluye la cláusula WHERE afectará a TODAS las tuplas de la tabla.**

**Ejemplo:** Actualizar el domicilio del alumno con nif 07345128H a: 'C\ Tormes, 2' de Badajoz (código de provincia 6 y código postal 06001' ).

```
UPDATE alumnos101
SET domicilio = 'C\ TORMES, 2',
    poblacion = 'BADAJOZ',
    cod_postal=6001 ,
    cod_prov = 6
WHERE nif= '07345128H';

1 fila actualizada.
```

## 6. Borrado de tuplas

**Sintaxis:**

```
DELETE
FROM tabla
[WHERE expresión]
```

**Notas**

- Se recomienda ejecutar la sentencia SELECT con la condición impuesta en el borrado y, una vez comprobado que la salida es lo que se desea borrar, únicamente se tendrá que sustituir SELECT por DELETE.
- Al igual que en la sentencia UPDATE si no incluimos la cláusula WHERE, la sentencia afectará a todas las tuplas de la tabla, y por tanto, serán todas borradas.

**Ejemplo:** Borrado del alumno con nif 07345128H matriculado en la titulación 101.

```
DELETE FROM alumnos101
WHERE nif='07345128H';
```

# ANEXO I Ejemplos de Funciones de tratamiento de cadenas

## *Unión de cadenas*

**Ejemplo:** Seleccionar los números de expediente y el nombre de los alumnos con su ciudad entre paréntesis

**Oracle**

```
SELECT num_exp, nombre ||' (' || poblacion || ')'
FROM practbd.alumnos;

   NUM_EXP NOMBRE||'('||POBLACION||')'
---------- ------------------------------------------------
         1 JUAN (BADAJOZ)
         2 PAULA (CACERES)
         3 JUAN PEDRO (CACERES)
         4 IRENE (CACERES)
         5 ALBERTO (BADAJOZ)
         6 RUBEN (CACERES)
```

## *Cambio de cabeceras de columnas*

Debe realizarse utilizando la palabra reservada AS aunque Oracle permite cambiar la cabecera de la columna sin utilizar AS, simplemente indicando el nombre a mostrar después de la expresión

**Ejemplo:** La consulta anterior, cambiando la cabecera de la segunda columna por Alumno

```
SELECT num_exp, nombre ||' (' || poblacion || ')' AS Alumno
FROM practbd.alumnos;

o

SELECT num_exp, nombre ||' (' || poblacion || ')' Alumno
FROM practbd.alumnos;



   NUM_EXP Alumno
---------- ------------------------------------------------
         1 JUAN (BADAJOZ)
         2 PAULA (CACERES)
         3 JUAN PEDRO (CACERES)
         4 IRENE (CACERES)
         5 ALBERTO (BADAJOZ)
         6 RUBEN (CACERES)
```

## *Aumento del tamaño de una columna*

**Ejemplo:** Seleccionar el número de expediente, el nombre y la nota media de los alumnos. Para los que no hayan finalizado sus estudios, poner como nota –1. Rellenar a puntos los campos.

**Oracle**

```
SELECT RPAD(num_exp,12,'.') Exp,
       RPAD(nombre,20,'.') Alumno,
       LPAD(NVL(nota_media,-1),12,'.') Nota
FROM practbd.alumnos;
```

```
EXP          ALUMNO               NOTA
------------ -------------------- ------------
1........... JUAN................ ..........-1
2........... PAULA............... ..........-1
3........... JUAN PEDRO.......... ..........-1
4........... IRENE............... ........6,75
5........... ALBERTO............. ........7,05
6........... RUBEN............... ..........-1
```

**LPAD**: Inserta caracteres a la izquierda con el mismo formato que RPAD

## *Borrado de caracteres*

**Ejemplo:** Seleccionar las poblaciones de residencia de los alumnos eliminando por la izquierda cualquier cadena ([CA])*

**Oracle**

```
SELECT DISTINCT LTRIM (poblacion,'CA')
FROM practbd.alumnos;

LTRIM(POBLACION,'CA')
------------------------
BADAJOZ
ERES
```

**RTRIM**: Elimina caracteres a la derecha.

## *Longitud de la cadena*

**LENGTH (cadena):** Nos devuelve el tamaño de la cadena actual.

**Ejemplo:** Seleccionar las poblaciones de los alumnos que tienen más de seis caracteres

```
SELECT DISTINCT poblacion
FROM practbd.alumnos
WHERE length(poblacion) > 6;

POBLACION
------------------------
BADAJOZ
CACERES
```

# Anexo II: Esquema sakila

Para el desarrollo de las prácticas se ha incluido dentro del esquema sakila de Oracle, una base de datos con información acerca de una tienda de alquiler de dvd de películas.

Inicialmente, dicha base de datos sólo cuenta con una tabla llamada FILM con los siguientes atributos:

| ATRIBUTO | TIPO | DESCRIPCION | NULO |
|---|---|---|---|
| FILM_ID | NUMBER(10) | Identificador de la película. | NO |
| TITLE | VARCHAR2(255) | Titulo. | NO |
| DESCRIPTION | CLOB | Trama. | SI |
| RELEASE_YEAR | NUMBER(4) | Año de lanzamiento. | SI |
| LANGUAGE_ID | NUMBER(10) | Código del idioma en la que se encuentra el dvd de la película. | NO |
| ORIGINAL_LANGUAGE_ID | NUMBER(10) | Código del idioma original de la película. | SI |
| RENTAL_DURATION | NUMBER(10) | Duración del alquiler del dvd en días. | NO |
| RENTAL_RATE | NUMBER(6,2) | Precio del alquiler del dvd en dólares. | NO |
| LENGTH | NUMBER(10) | Duración de la película en minutos. | SI |
| REPLACEMENT_COST | NUMBER(6,2) | Coste de reemplazo del dvd. | NO |
| RATING | VARCHAR2(5) | Clasificación parental, | SI |
| SPECIAL_FEATURES | VARCHAR2(54) | Características especiales del dvd. | SI |
| LAST_UPDATE | DATE | Ultima actualización de los datos del dvd. | NO |

En las siguientes prácticas se incluirán el resto de tablas que forman la base de datos, así como las relaciones entre ellas y sus restricciones.

# Anexo III: Ejercicios con la tabla FILM del esquema sakila

1. Crear una tabla denominada peliculas_tp con toda la información de la tabla film cuya clasificación (rating) sea para todos los públicos, sin supervisión parental (G) o con ella (PG). Borrar la tabla previamente por si existiese.

   **Realizar el resto de ejercicios usando la tabla peliculas_tp que hemos creado en el ejercicio anterior.**

2. Obtener el identificador de la película, el título y la duración de las películas cuya duración supere los 180 minutos y cuyo periodo de alquiler esté entre 2 y 4 días, ordenando de mayor a menor duración y, en caso de igualdad, alfabéticamente por el título.

```
ID   TITULO               DURACION
---- -------------------- --------
991  WORST BANGER         185
50   BAKED CLEOPATRA      182
```

3. Obtener los identificadores y los títulos de las películas que contienen la palabra CAT (gato) dentro del título en cualquier posición del mismo.

```
PELICULA
--------------------------------------------------
127 - CAT CONEHEADS
```

4. Seleccionar el título y la duración de las películas con identificador menor que 50 y cuyas características especiales sólo incluyen la palabra 'Trailers'.

```
TITLE            DURACION
---------------- ------------------
ANGELS LIFE      1 HORAS 14 MINUTOS
ARMAGEDDON LOST  1 HORAS 39 MINUTOS
```

5. Seleccionar en el mínimo número de consultas posibles, la duración mínima, máxima y el número de películas, el coste total de reemplazar todas las películas, así como el nombre y el número de todas las clasificaciones existentes.

```
MINIMO MAXIMO PELICULAS COSTE_REEMPLAZO CLASIFICACIONES
------ ------ --------- --------------- --------------
    46    185       372         7260,28              2

RATING
------
PG
G
```

6. Dar de alta una nueva película generando su identificador de película de una secuencia *spelicula*, que debes crear previamente, empezando en el número 1001 e incrementándose de 1 en 1.
7. Dar de alta una nueva película con los mismos datos que la anterior excepto su identificador y el título que será el mismo pero añadiéndole un 2.
8. Actualizar la duración a 90 y el idioma original a 1 para las películas que no tienen relleno ninguno de los dos campos.
9. Incrementar en 1 dólar el precio de alquiler de las películas clasificadas como 'PG' y cuya duración del alquiler esté entre 4 y 5 días.
10. Eliminar las películas clasificadas como 'PG' que no tengan relleno la duración de la película o el año de lanzamiento.