1. **Escribe un procedimiento que dado el título de una serie muestre el código que le corresponde siempre y cuando no haya más de uno.**

```
CREATE OR REPLACE PROCEDURE BUSCAR_CODIGO_SERIE (TITULO SERIE.SERIE_TITULO%TYPE) IS

 CODIGO SERIE.SERIE_ID%TYPE;
 CUENTA NUMBER(1);

BEGIN
 SELECT COUNT(SERIE_TITULO) INTO CUENTA
 FROM SERIE
 WHERE SERIE_TITULO=TITULO;

  IF CUENTA=1 THEN
   SELECT SERIE_ID INTO CODIGO
   FROM SERIE
   WHERE SERIE_TITULO=TITULO;
   DBMS_OUTPUT.PUT_LINE('Su código es '||CODIGO);
  ELSE
   DBMS_OUTPUT.PUT_LINE('El titulo dado no existe o hay mas de 1');
  END IF;
END;
```

- **Conversión a función del ejercicio 1:**

```
create or replace FUNCTION BUSCARCODIGOSERIE_FUN ( TITULO SERIE.SERIE_TITULO%TYPE)
RETURN SERIE.SERIE_ID%TYPE
IS

  CONTADOR NUMBER(1);
  CODIGO SERIE.SERIE_ID%TYPE;

BEGIN

  SELECT COUNT(*) INTO CONTADOR
  FROM SERIE
  WHERE SERIE_TITULO=TITULO;

  IF CONTADOR=1 THEN
     SELECT SERIE_ID INTO CODIGO
     FROM SERIE
     WHERE SERIE_TITULO=TITULO;

     RETURN CODIGO;
  ELSE

     RETURN NULL;
  END IF;
END;

SELECT BUSCARCODIGOSERIE_FUN('Borgen')
FROM DUAL;
```

2. **Escribe un procedimiento que dado el código de una serie muestre el total de capítulos.**

```
CREATE OR REPLACE PROCEDURE CONTARCAPITULOS ( CODIGO CAPITULO.SERIE_ID%TYPE) IS

        CONTADOR NUMBER(1);

  BEGIN

    SELECT COUNT(*) INTO CONTADOR
    FROM CAPITULO
   WHERE SERIE_ID=CODIGO;

     DBMS_OUTPUT.PUT_LINE('HAY '||CONTADOR||' CAPITULOS PARA ESE CODIGO');

END;



EXECUTE CONTARCAPITULOS('BRGN');
```

- **Conversión a función del ejercicio 2:**

```
create or replace FUNCTION  CONTARCAPITULOS_FUNCT ( CODIGO CAPITULO.SERIE_ID%TYPE)
RETURN NUMBER
IS

    CONTADOR NUMBER(1);

  BEGIN

    SELECT COUNT(*) INTO CONTADOR
    FROM CAPITULO
    WHERE SERIE_ID=CODIGO;

    RETURN CONTADOR;

  END;

  SELECT CONTARCAPITULOS_FUNCT('BRGN') FROM DUAL;
```

Utilizar la función correspondiente al ejercicio2 para rellenar la columna CAPÍTULOS de la tabla SERIE.

```
UPDATE SERIE
SET CAPITULOS = (SELECT COUNT(*)
            FROM CAPITULO
            WHERE CAPITULO.SERIE_ID=SERIE.SERIE_ID):

UPDATE SERIE
SET CAPITULOS = CONTARCAPITULOS_FUNC(SERIE_ID)
```

3. **Escribe un procedimiento que dado el nombre de un autor, muestre el número de personajes que interpreta.**

```
CREATE OR REPLACE PROCEDURE NUMERO_PERSONAJES_INTERPRETA (NOMBRE
ACTOR.ACTOR_NOMBRE%TYPE) IS

CUENTA NUMBER(1);
PERSONAJES NUMBER(2);

BEGIN

 SELECT COUNT(ACTOR_NOMBRE) INTO CUENTA
 FROM ACTOR
 WHERE ACTOR_NOMBRE=NOMBRE;

 IF CUENTA=1 THEN
   SELECT COUNT(*) INTO PERSONAJES
   FROM REPARTO JOIN ACTOR USING (ACTOR_ID)
   WHERE ACTOR_NOMBRE=NOMBRE;

   DBMS_OUTPUT.PUT_LINE(NOMBRE||' interpreta a '||PERSONAJES||' personajes.');
  ELSE
   DBMS_OUTPUT.PUT_LINE('El autor no existe.');
  END IF;
END;
```

Conversión a función del ejercicio 3

```
CREATE OR REPLACE FUNCTION NUMPERSONAJE_FUNC(NOMBREACTOR ACTOR.ACTOR_NOMBRE%TYPE)
RETURN NUMBER
IS
  CONTADOR NUMBER(1);
  TOTALPERSONAJE NUMBER(2);
BEGIN
  SELECT COUNT(*)INTO CONTADOR
  FROM ACTOR
  WHERE ACTOR_NOMBRE=NOMBREACTOR;

    IF CONTADOR=1 THEN
      SELECT COUNT(*)INTO TOTALPERSONAJE
      FROM REPARTO JOIN ACTOR USING(ACTOR_ID)
      WHERE ACTOR_NOMBRE=NOMBREACTOR;
      RETURN TOTALPERSONAJE;
    ELSE
      RETURN 0;
    END IF;
END;

SELECT NUMPERSONAJE_FUNC('Bryan Cranston') FROM DUAL;
```

- Dado un código de departamento, escribe una función que devuelva la suma salarial de sus empleados.

```
CREATE OR REPLACE FUNCTION SUMASALARIAL_FUNC ( CODIGO DEPARTMENTS.DEPARTMENT_ID%TYPE)
RETURN NUMBER
IS
CONTADOR NUMBER(2);
SUMASALARIAL NUMBER(9,2);
BEGIN


   SELECT SUM(SALARY) INTO SUMASALARIAL
   FROM EMPLOYEES
   WHERE DEPARTMENT_ID=CODIGO;
   RETURN SUMASALARIAL;

END;

SELECT SUMASALARIAL_FUNC(100) FROM DUAL
```

1. Usando la función anterior, actualiza la columna SUMASALARIAL de la tabla DEPARTMENTS.

```
UPDATE DEPARTMENTS
SET SUMASALARIAL=SUMASALARIAL_FUNC(DEPARTMENT_ID);
```

- Escribe un procedimiento que reciba un nombre de departamento y calcule la suma salarial de los empleados que trabajan en él para mostrarla por pantalla.

```
CREATE OR REPLACE PROCEDURE SUMASALARIAL_PROC (NOMBREDEP
DEPARTMENTS.DEPARTMENT_NAME%TYPE) IS
SUMA_SALARY NUMBER(9,2);
CODIGO DEPARTMENTS.DEPARTMENT_ID%TYPE;
CONTADOR NUMBER(2);

BEGIN

 SELECT COUNT(*) INTO CONTADOR
 FROM DEPARTMENTS
 WHERE DEPARTMENT_NAME=NOMBREDEP;

 IF CONTADOR=0 THEN
   DBMS_OUTPUT.PUT_LINE('El departamento no existe o no hay suma para él.');
 ELSE

   SELECT DEPARTMENT_ID INTO CODIGO
   FROM DEPARTMENTS
   WHERE DEPARTMENT_NAME=NOMBREDEP;
```

```
   SUMA_SALARY:=SUMASALARIAL_FUNC(CODIGO);

   DBMS_OUTPUT.PUT_LINE('La suma salarial es '||SUMA_SALARY||' €.');
 END IF;
END;


EXECUTE SUMASALARIAL_PROC('Sales');
```

PL/SQL SOBRE PUBLICACIONES

1. **Escribe un procedimiento que dado un título de libro, muestre el nombre de la editorial que lo publica.**

```
CREATE OR REPLACE PROCEDURE EDITORIALPUB (TITULO TITLE.TITLE%TYPE) IS

CONTADOR NUMBER(1);
EDITORIAL PUBLISHER.PUB_NAME%TYPE;

BEGIN

SELECT COUNT(*) INTO CONTADOR
FROM TITLE
WHERE TITLE=TITULO;

IF CONTADOR=0 THEN
DBMS_OUTPUT.PUT_LINE('El título dado no existe');

ELSE
SELECT PUB_NAME INTO EDITORIAL
FROM PUBLISHER JOIN TITLE USING (PUB_ID)
WHERE TITLE=TITULO;

DBMS_OUTPUT.PUT_LINE('El nombre de la editorial es '||EDITORIAL);

END IF;
END;
```

Versión 2:

```
create or replace PROCEDURE EDITORIALPUB (TITULO TITLE.TITLE%TYPE) IS

EDITORIAL PUBLISHER.PUB_NAME%TYPE;

BEGIN

SELECT PUB_NAME INTO EDITORIAL
FROM PUBLISHER JOIN TITLE USING (PUB_ID)
WHERE TITLE=TITULO;

DBMS_OUTPUT.PUT_LINE('El nombre de la editorial es '||EDITORIAL);

EXCEPTION
  WHEN NO_DATA_FOUND THEN
```

```
   DBMS_OUTPUT.PUT_LINE('El título dado no existe');
END;
```

2. **Escribe una función que dado un título de libro, devuelva el nombre de la editorial que lo publica.**

```
CREATE OR REPLACE FUNCTION EDITORIALPUB_FUNC (TITULO TITLE.TITLE%TYPE)

RETURN VARCHAR2
IS
CONTADOR NUMBER(1);
EDITORIAL PUBLISHER.PUB_NAME%TYPE;
BEGIN

SELECT COUNT(*) INTO CONTADOR
FROM TITLE
WHERE TITLE=TITULO;

IF CONTADOR=0 THEN
RETURN 0;

ELSE
SELECT PUB_NAME INTO EDITORIAL
FROM PUBLISHER JOIN TITLE USING (PUB_ID)
WHERE TITLE=TITULO;
RETURN EDITORIAL;

END IF;
END;

SELECT EDITORIALPUB_FUNC('Secrets of Silicon Valley')
FROM DUAL
```

Versión 2:

```
create or replace FUNCTION EDITORIALPUB_FUNC (TITULO TITLE.TITLE%TYPE)

RETURN VARCHAR2
IS

EDITORIAL PUBLISHER.PUB_NAME%TYPE;
BEGIN

SELECT PUB_NAME INTO EDITORIAL
FROM PUBLISHER JOIN TITLE USING (PUB_ID)
WHERE TITLE=TITULO;

RETURN EDITORIAL;

EXCEPTION

 WHEN NO_DATA_FOUND THEN
   RETURN 'NO SE PUEDE BUSCAR LA EDITORIAL';

END;
```

```
SELECT TITLE, EDITORIALPUB_FUNC(TITLE) FROM TITLE;
```

3. **Escribe una función que dado un título de libro, devuelva su código.**

```
CREATE OR REPLACE FUNCTION CODIGO_FUNC (TITULO TITLE.TITLE%TYPE)

RETURN VARCHAR2
IS
CONTADOR NUMBER(1);
CODIGO TITLE.TITLE_ID%TYPE;
BEGIN

SELECT COUNT(*) INTO CONTADOR
FROM TITLE
WHERE TITLE=TITULO;

IF CONTADOR=0 THEN
RETURN 0;

ELSE
SELECT TITLE_ID INTO CODIGO
FROM TITLE
WHERE TITLE=TITULO;
RETURN CODIGO;

END IF;
END;

SELECT CODIGO_FUNC('Secrets of Silicon Valley')
FROM DUAL
```

4. **Escribe una función que dado un código de libro, devuelva el dinero que ganan sus autores en un año:**

ytd_sales*Price*Royalty

```
CREATE OR REPLACE FUNCTION DINEROAUT_FUNC (CODLIBRO TITLE.TITLE_ID%TYPE)

RETURN NUMBER
IS
CONTADOR NUMBER(1);
DINEROAUT NUMBER(8);
BEGIN

SELECT COUNT(*) INTO CONTADOR
```

```
  FROM TITLE
  WHERE TITLE_ID=CODLIBRO;

  IF CONTADOR=0 THEN
  RETURN 0;

  ELSE
  SELECT YTD_SALE*PRICE*(ROYALTY/100) INTO DINEROAUT
  FROM TITLE
  WHERE TITLE_ID=CODLIBRO;
  RETURN DINEROAUT;

  END IF;
  END;

  SELECT DINEROAUT_FUNC('PC8888')
  FROM DUAL
```

5. **Escribe un procedimiento que dado un título de libro muestre el nombre de sus autores y el dinero que han ganado cada uno.**

```
CREATE OR REPLACE PROCEDURE AUTORES_DINERO_LIBRO2 (TITULO TITLE.TITLE%TYPE) IS

CURSOR AUTORES IS
SELECT TITLE_ID, AU_FNAME, AU_LNAME, ROYALTYPER
FROM AUTHOR JOIN TITLEAUTHOR USING (AU_ID)
        JOIN TITLE USING (TITLE_ID)
WHERE TITLE=TITULO;

CONTADOR NUMBER(1):=0;

BEGIN

FOR REGISTRO IN AUTORES LOOP

   DBMS_OUTPUT.PUT_LINE('AUTOR: '||REGISTRO.AU_FNAME||' '||REGISTRO.AU_LNAME);
   DBMS_OUTPUT.PUT_LINE('GANA:
'||DINEROAUT_FUNC(REGISTRO.TITLE_ID)*REGISTRO.ROYALTYPER/100);
   CONTADOR:=CONTADOR+1;
 END LOOP;

 IF CONTADOR=0 THEN
  DBMS_OUTPUT.PUT_LINE('NO SE HAN ENCONTRADO AUTORES');
 END IF;
END;
EXECUTE AUTORES_DINERO_LIBRO2('The Busy Executive''s Database Guide');
```

6. **Escribe un procedimiento que dado el nombre de una editorial, muestra un listado con los títulos de los libros que ha publicado.**

```
CREATE OR REPLACE PROCEDURE TITULOS_LIBROS (EDITORIAL PUBLISHER.PUB_NAME%TYPE) IS

CURSOR TITULOS IS
SELECT TITLE
FROM TITLE JOIN PUBLISHER USING (PUB_ID)
WHERE PUB_NAME=EDITORIAL;

CONTADOR NUMBER(1);
BEGIN

SELECT COUNT(*) INTO CONTADOR
FROM TITLE JOIN PUBLISHER USING (PUB_ID)
WHERE PUB_NAME=EDITORIAL;

IF CONTADOR=0 THEN
  DBMS_OUTPUT.PUT_LINE('La editorial no tiene títulos o no existe');
ELSE

FOR REGISTRO IN TITULOS LOOP

    DBMS_OUTPUT.PUT_LINE(REGISTRO.TITLE);

 END LOOP;

END IF;

END;

EXECUTE TITULOS_LIBROS('New Moon Books');
```

7. **Añade al procedimiento anterior la visualización de los autores de cada libro y el dinero que han ganado con sus ventas.**

```
CREATE OR REPLACE PROCEDURE TITULOS_LIBROS2 (EDITORIAL PUBLISHER.PUB_NAME%TYPE) IS

CURSOR TITULOS IS
SELECT TITLE
FROM TITLE JOIN PUBLISHER USING (PUB_ID)
WHERE PUB_NAME=EDITORIAL;

CONTADOR NUMBER(1);
BEGIN

SELECT COUNT(*) INTO CONTADOR
FROM TITLE JOIN PUBLISHER USING (PUB_ID)
WHERE PUB_NAME=EDITORIAL;

IF CONTADOR=0 THEN
```

```
DBMS_OUTPUT.PUT_LINE('La editorial no tiene titulos o no existe.');
ELSE

FOR REGISTRO IN TITULOS LOOP

   DBMS_OUTPUT.PUT_LINE(' ');
   DBMS_OUTPUT.PUT_LINE(' ');
   DBMS_OUTPUT.PUT_LINE('TITULO DE LIBRO DE DICHA EDITORIAL: '||REGISTRO.TITLE);
   DBMS_OUTPUT.PUT_LINE(' ');
   AUTORES_DINERO_LIBRO2(REGISTRO.TITLE);

 END LOOP;
 END IF;
END;
```

8. **Escribe un procedimiento de incremento de ventas para un título de libro dado y un valor adicional de ventas.**

```
CREATE OR REPLACE PROCEDURE INCR_VENTAS (TITULO TITLE.TITLE%TYPE, VALOR TITLE.YTD_SALE%TYPE)
IS

CONTADOR NUMBER(1);
BEGIN

SELECT COUNT(*) INTO CONTADOR
FROM TITLE
WHERE TITLE=TITULO;

IF CONTADOR=0 THEN
DBMS_OUTPUT.PUT_LINE('La editorial no tiene titulos o no existe.');
ELSE

UPDATE TITLE
SET YTD_SALE = NVL(YTD_SALE, 0) +VALOR WHERE TITLE=TITULO;

END IF;
END;

EXECUTE INCR_VENTAS('Secrets of Silicon Valley', 4000)
```

9. **Escribe una función que dado el nombre de una editorial, calcule y devuelva el total de libros publicados.**

```
CREATE OR REPLACE FUNCTION TOTAL_LIBROS_FUNC (EDITORIAL PUBLISHER.PUB_NAME%TYPE)
RETURN VARCHAR2
IS
CONTADOR NUMBER(1);
BEGIN

SELECT COUNT(*) INTO CONTADOR
FROM TITLE JOIN PUBLISHER USING (PUB_ID)
WHERE PUB_NAME=EDITORIAL;
```

```
RETURN CONTADOR;
END;

SELECT TOTAL_LIBROS_FUNC('Binnet and Hardley')
FROM DUAL
```

10. **Añade la columna PUBLICADOS a la taba PUBLISHERS y rellénala con los datos que te devuelve la función anterior.**

```
ALTER TABLE PUBLISHER ADD PUBLICADOS NUMBER(2)

UPDATE PUBLISHER
SET PUBLICADOS = (SELECT COUNT(*)
            FROM TITLE

        WHERE PUB_ID=PUBLISHER.P
```

EJERCICIOS PL EN SAKILA

1. **Escribe una función que dado un código de alquiler (rental_id) devuelva el título que corresponde a la película alquilada.**

```
CREATE OR REPLACE FUNCTION TITULO_PELI_ALQ_FUNC (CODIGO RENTAL.RENTAL_ID%TYPE)
RETURN VARCHAR2
IS
CONTADOR NUMBER(1);
TITULO FILM.TITLE%TYPE;
BEGIN

SELECT COUNT(*) INTO CONTADOR
FROM RENTAL
WHERE RENTAL_ID=CODIGO;

IF CONTADOR=0 THEN
RETURN 0;
ELSE
SELECT TITLE INTO TITULO
FROM FILM JOIN INVENTORY USING (FILM_ID)
      JOIN RENTAL USING (INVENTORY_ID)
WHERE RENTAL_ID=CODIGO;
RETURN TITULO;
END IF;
END;

SELECT TITULO_PELI_ALQ_FUNC(1)
FROM DUAL
```

2. **Escribe una función que dado el nombre de un cliente devuelva su código.**

```
CREATE OR REPLACE FUNCTION CODIGO_CLIENTE_FUNC (NOMBRE CUSTOMER.FIRST_NAME%TYPE,
APELLIDO CUSTOMER.LAST_NAME%TYPE)
```

```
RETURN NUMBER
IS
CONTADOR NUMBER(1);
CODIGO FILM.TITLE%TYPE;
BEGIN

SELECT COUNT(*) INTO CONTADOR
FROM CUSTOMER
WHERE FIRST_NAME=NOMBRE AND LAST_NAME=APELLIDO;

IF CONTADOR=0 THEN
RETURN 0;
ELSE
SELECT CUSTOMER_ID INTO CODIGO
FROM CUSTOMER
WHERE FIRST_NAME=NOMBRE AND LAST_NAME=APELLIDO;
RETURN CODIGO;
END IF;
END;

SELECT CODIGO_CLIENTE_FUNC('MARY', 'SMITH')
FROM DUAL
```

3. **Escribe un procedimiento que dado el nombre de un cliente muestre en pantalla un listado con los títulos de películas que ha alquilado. Emite también al final un mensaje que indique el número total de películas alquiladas. (NOTA: Puedes utilizar las funciones de los ejercicios anteriores).**

```
CREATE OR REPLACE PROCEDURE EJER3 ( NOMBRE CUSTOMER.FIRST_NAME%TYPE,
                        APELLIDO CUSTOMER.LAST_NAME%TYPE) IS
CURSOR PELICULAS IS
SELECT TITLE
FROM FILM JOIN INVENTORY USING (FILM_ID)
        JOIN RENTAL USING(INVENTORY_ID)
        JOIN CUSTOMER USING (CUSTOMER_ID)
WHERE FIRST_NAME=NOMBRE AND LAST_NAME=APELLIDO;

CONTADOR NUMBER(2):=0;

BEGIN

  FOR FILA IN PELICULAS LOOP

     DBMS_OUTPUT.PUT_LINE(FILA.TITLE);

     CONTADOR:=CONTADOR+1;
  END LOOP;

  DBMS_OUTPUT.PUT_LINE('TOTAL DE PELÍCULAS ALQUILADAS: '||CONTADOR);
END;

SET SERVEROUTPUT ON

EXECUTE EJER3('MARY', 'SMITH');
```

```
CREATE OR REPLACE PROCEDURE EJER3_V2 ( NOMBRE CUSTOMER.FIRST_NAME%TYPE,
                        APELLIDO CUSTOMER.LAST_NAME%TYPE) IS
CURSOR PELICULAS IS
SELECT FUNC1(RENTAL_ID) PELICULA
FROM RENTAL
WHERE CUSTOMER_ID=FUNC2(NOMBRE, APELLIDO);

CONTADOR NUMBER(2):=0;

BEGIN

  FOR FILA IN PELICULAS LOOP

    DBMS_OUTPUT.PUT_LINE(FILA.PELICULA);

    CONTADOR:=CONTADOR+1;
  END LOOP;

  DBMS_OUTPUT.PUT_LINE('TOTAL DE PELÍCULAS ALQUILADAS: '||CONTADOR);
END;

SET SERVEROUTPUT ON

EXECUTE EJER3_V2('MARY', 'SMITH');
```

4. **Escribe un procedimiento que dado un título de película, muestre un listado de actores que participan en ella.**

```
CREATE OR REPLACE PROCEDURE EJER4 ( TITULO FILM.TITLE%TYPE) IS
CURSOR ACTORES IS
  SELECT FIRST_NAME, LAST_NAME
  FROM ACTORES JOIN FILM_ACTOR USING (ACTOR_ID)
          JOIN FILM USING (FILM_ID)
  WHERE TITLE=TITULO;

BEGIN

  FOR REG IN ACTORES LOOP
    DBMS_OUTPUT.PUT_LINE(REG.FIRST_NAME||' '||REG.LAST_NAME);
  END LOOP;
END;

EXECUTE EJER4('ACADEMY DINOSAUR');
```

PREXAMEN:

5. **Escribe un procedimiento que dado un título de película, muestre un listado de actores que participan en ella.**

```
CREATE OR REPLACE PROCEDURE EJER4 ( TITULO FILM.TITLE%TYPE) IS
CURSOR ACTORES IS
  SELECT FIRST_NAME, LAST_NAME
  FROM ACTORES JOIN FILM_ACTOR USING (ACTOR_ID)
            JOIN FILM USING (FILM_ID)
  WHERE TITLE=TITULO;

BEGIN

  FOR REG IN ACTORES LOOP
    DBMS_OUTPUT.PUT_LINE(REG.FIRST_NAME||' '||REG.LAST_NAME);
  END LOOP;
END;

EXECUTE EJER4('ACADEMY DINOSAUR');

UPDATE ACTORES
SET NUMPELIS = (SELECT COUNT(*)
            FROM FILM_ACTOR
            WHERE ACTOR_ID= ACTORES.ACTOR_ID);
```

/*CREAR UN TRIGGER QUE MANTENGA ACTUALIZADO EL CAMPO NUMPELIS DE LA

TABLA ACTORES*/

```
CREATE OR REPLACE TRIGGER T2 AFTER INSERT OR DELETE ON FILM_ACTOR FOR EACH ROW
BEGIN

 IF DELETING THEN
    UPDATE ACTOR SET NUMPELIS=NUMPLEIS-1
    WHERE ACTOR_ID=:OLD.ACTOR_ID;
 ELSE
     UPDATE ACTOR SET NUMPELIS=NUMPLEIS+1
    WHERE ACTOR_ID=:NEW.ACTOR_ID;
 END IF;
END;
```

/*EL REPLACEMENT_COST NO PUEDE SER NUNCA 20 VECES MAYO QUE EL PRECIO

DE UN ALQUILER, ELEVANDO ERROR*/

```
CREATE OR REPLACE TRIGGER T3 BEFORE INSERT OR UPDATE ON FILM FOR EACH ROW
WHEN ( NEW.REPLACEMENT_COST > NEW.RENTAL_RATE *20)
BEGIN
    RAISE_APPLICATION_ERROR(-20100, 'NO SE PUEDA SUPERAR 20 ...');

END;
```

```
UPDATE ACTORES
SET NUMPELIS = (SELECT COUNT(*)
        FROM FILM_ACTOR
        WHERE ACTOR_ID=ACTORES.ACTOR_ID);
```

```
CREATE OR REPLACE TRIGGER T2 BEFORE UPDATE OR INSERT ON FILM
FOR EACH ROW
WHEN (NEW.REPLACEMENT_COST > NEW.RENTAL_RATE*20)
BEGIN
RAISE_APPLICATION_ERROR(-20100,'NO SE PUEDE SUPERAR 20...');
END;
```