FUNCIONA

***WITH interpretes AS (SELECT DISTINCT name FROM PERFORMERS), conciertos\_interpretes AS (SELECT DISTINCT C.performer, I.name FROM CONCERTS C INNER JOIN interpretes I ON (C.performer = I.name)), performer\_performances AS (SELECT DISTINCT CI.name, P.songwriter FROM PERFORMANCES P INNER JOIN conciertos\_interpretes CI ON (P.performer=CI.performer)), interpreted\_songs AS (SELECT DISTINCT PP.name, S.writer, S.cowriter FROM SONGS S INNER JOIN performer\_performances PP ON (PP.songwriter=S.writer)), tracked\_songs AS (SELECT DISTINCT INS.name FROM TRACKS T INNER JOIN interpreted\_songs INS ON(INS.writer = T.writer OR INS.cowriter = T.writer)) SELECT DISTINCT name AS “nombre interprete” FROM tracked\_songs ;***

– 542 / 695 filas seleccionadas

| I ≡ PERFORMERS (intérpretes) → letra |
| --- |
| C ≡ CONCERTS (conciertos) → letra |
| P ≡ PERFORMANCES (interpretaciones) → letra performer |
| G ≡ INVOLVEMENT (grupo) → band letra y name número |
| M ≡ MUSICIANS (músicos) → passport número |
| S ≡ SONGS (canciones) → número |
| T ≡ TRACKS (pistas) → número |
| P ≡ PERFORMANCES (interpretaciones) → número songwriter |

consulta 1

* de los I quiero sacar el NAME
* los conciertos que son de ese interprete, de C quiero PERFORMER = I.NAME
* las actuaciones que son de ese concierto, de P quiero P.PERFORMER = C.PERFORMER
* los intérpretes que forman parte de algún grupo, quiero I.NAME == G.BAND
* de los de un grupo quiero buscar sus pasaportes G.NAME = M.PASSPORT
* de esos músicos M.PASSPORT = S.WRITER OR M.PASSPORT = S.COWRITER WHEN COWRITER IS NOT NULL
* de las tracks quiero que T.TITLE = S.TITLE AND T.WRITER = S.WRITER
* de las actuaciones queremos que P.SONGWRITER = S.WRITER AND P.SONGTITLE = S.TITLE

devuelve -> I..NAME, % TRACKS QUE SON PROPIAS, % INT EN CONCIERTOS QUE SON PROPIAS

consulta 2

P ≡ PERFORMANCES (interpretaciones) → letra performer

S ≡ SONGS (canciones) → número

T ≡ TRACKS (pistas) → número

P ≡ PERFORMANCES (interpretaciones) → número songwriteR

T ≡ TRACKS (pistas) → RECORDING\_DATE DATE

P ≡ PERFORMANCES (interpretaciones) → WHEN DATE

S ≡ SONGS (canciones) → número

* las T.TILE = S.TITLE AND T.WRITER = S.WRITER OR T.WRITER = S.COWRITER WHEN COWRITER IS NOT NULL
* P.SONGTITLE = S.TITLE AND S.WRITER = P.SONGWRITER OR S.COWRITER = P.SONWRITER WHEN COWRITER IS NOT NULL

WITH interpretes AS (SELECT DISTINCT PER.name FROM performers PER), conciertos\_interpretes AS ( SELECT DISTINCT C.performer, I.name FROM CONCERTS C INNER JOIN interpretes I ON (C.performer = I.name)), performer\_performances AS (SELECT DISTINCT P.performer, CI.name, P.songwriter, P.songtitle FROM PERFORMANCES P INNER JOIN conciertos\_interpretes CI ON (P.performer = CI.performer)), interpreted\_songs AS (SELECT DISTINCT PP.performer, PP.songtitle, PP.songwriter, PP.name FROM SONGS S INNER JOIN performer\_performances PP ON ((PP.songwriter = S.writer OR PP.songwriter = S.cowriter ) AND PP.songtitle = S.title)), interprete\_grupo AS (SELECT DISTINCT G.musician, G.band FROM INVOLVEMENT G INNER JOIN interpretes I ON (I.name = G.band)), musico\_grupo AS (SELECT DISTINCT IG.musician, IG.band, M.passport FROM MUSICIANS M INNER JOIN interprete\_grupo IG ON (M.passport = IG.musician)), song\_writer AS (SELECT DISTINCT S.writer, S.cowriter, MG.musician, MG.band, MG.passport, S.title FROM SONGS S INNER JOIN musico\_grupo MG ON (S.writer = MG.passport OR S.cowriter = MG.passport AND MG.musician = MG.passport)), pistas\_canciones AS (SELECT DISTINCT T.writer, T.title, SW.musician, SW.band, SW.passport, SW.writer, SW.cowriter FROM TRACKS T INNER JOIN song\_writer SW ON (T.writer = SW.writer OR T.writer = SW.cowriter AND SW.title = T.title)), interpretaciones\_canc\_prop AS (SELECT DISTINCT INS.name, INS.performer, INS.songtitle, INS.songwriter, PC.musician, PC.band, PC.passport FROM pistas\_canciones PC INNER JOIN interpreted\_songs INS ON (PC.band = INS.name)) SELECT DISTINCT interpretaciones\_canc\_prop.name AS "nombre del interprete" FROM interpretaciones\_canc\_prop;

WITH interpretes AS ( SELECT DISTINCT name FROM performers), **#SQ1**

conciertos\_interpretes AS ( SELECT DISTINCT C.performer, I.name FROM CONCERTS C INNER JOIN interpretes I ON ( C.performer = I.name )), **#SQ2**

performer\_performances AS ( SELECT DISTINCT P.performer, CI.name, P.songwriter, P.songtitle FROM PERFORMANCES P INNER JOIN conciertos\_interprete CI ON (P.performer = CI.performer) ), **#SQ3**

interpreted\_songs AS ( SELECT DISTINCT PP.performer, PP.songtitle, PP.songwriter, PP.name FROM SONGS S INNER JOIN performer\_performances PP ON ( ( PP.songwriter = S.writer OR PP.songwriter = S.cowriter ) AND PP.songtitle = S.title ) ), **#SQ8**

interprete\_grupo AS (SELECT DISTINCT G.name, G.band FROM INVOLVEMENT G INNER JOIN interprete I ON ( I.name = G.band ), **#SQ4**

musico\_grupo AS (SELECT DISTINCT IG.name, IG.band, M.passport FROM MUSICIANS M INNER JOIN interprete\_grupo IG ON ( M.passport = IG.name) ), **#SQ5**

song\_writer AS (SELECT DISTINCT S.writer, S.cowriter, MG.name, MG.band, MG.passport FROM SONGS S INNER JOIN musico\_grupo MG ON ( S.writer = MG.passport OR S.cowriter = MG.passport ) ), **#SQ6**

pistas\_canciones AS (SELECT DISTINCT T.writer, T.title, SW.name, SW.band, SW.passport, SW.writer, SW.cowriter FROM TRACKS T INNER JOIN song\_writer SW ON ( ( T.writer = SW.writer OR T.writer = SW.cowriter ) AND SW.title = T.title) ), **#SQ7**

interpretaciones\_canc\_prop AS (SELECT DISTINCT PP.name, PP.performer, PP.songtitle, PP.songwriter, SW.name, SW.band, SW.passport FROM pistas\_canciones PC INNER JOIN interpreted\_song IS ON ( SW.band = PP.name AND SW.name = SW.passport ) ), **#SQ9**

porcentaje\_pistas\_grabadas AS (SELECT DISTINCT SW.name, porcentaje\_pp FROM pistas\_canciones SW WHERE(porcentaje\_pp = count(pistas\_canciones) / count(\*) from T \* 100 ), **#SQ10**

porcentaje\_interpretaciones\_propias AS (SELECT DISTINCT IG.name, porcentaje\_icp FROM interpreted\_songs IG WHERE(porcentaje\_icp = count(pistas\_canciones) / count(\*) from P \* 100 ) **#SQ11**

WITH

interpretes AS (SELECT DISTINCT name FROM PERFORMERS),

conciertos\_interpretes AS (SELECT DISTINCT C.performer, I.name FROM CONCERTS C INNER JOIN interpretes I ON (C.performer = I.name)),

performer\_performances AS (SELECT DISTINCT CI.name, P.songwriter FROM PERFORMANCES P INNER JOIN conciertos\_interpretes CI ON (P.performer=CI.performer)),

interpreted\_songs AS (SELECT DISTINCT PP.name, S.writer, S.cowriter FROM SONGS S INNER JOIN performer\_performances PP ON (PP.songwriter=S.writer)),

tracked\_songs AS (SELECT DISTINCT INS.name, COUNT(\*) AS num\_tracks, T.total\_tracks, (COUNT(\*) / T.total\_tracks \* 100) AS porcentaje\_pp

FROM TRACKS T

INNER JOIN interpreted\_songs INS ON (INS.writer = T.writer OR INS.cowriter = T.writer)

GROUP BY INS.name, T.total\_tracks)

SELECT DISTINCT name, porcentaje\_pp

FROM tracked\_songs;

WITH interpretes AS (SELECT DISTINCT name FROM PERFORMERS), conciertos\_interpretes AS (SELECT DISTINCT C.performer, I.name FROM CONCERTS C INNER JOIN interpretes I ON (C.performer = I.name)), performer\_performances AS (SELECT DISTINCT CI.name, P.songwriter FROM PERFORMANCES P INNER JOIN conciertos\_interpretes CI ON (P.performer=CI.performer)), interpreted\_songs AS (SELECT DISTINCT PP.name, S.writer, S.cowriter FROM SONGS S INNER JOIN performer\_performances PP ON (PP.songwriter=S.writer)),tracked\_songs AS (SELECT DISTINCT INS.name FROM TRACKS T INNER JOIN interpreted\_songs INS ON(INS.writer = T.writer OR INS.cowriter = T.writer)),porcentaje\_pistas\_grabadas AS (SELECT DISTINCT TS.name, (COUNT(\*)/ (SELECT COUNT(\*) FROM TRACKS)\* 100) AS porcentaje\_pp FROM tracked\_songs TS GROUP BY TS.name), porcentaje\_interpretaciones\_propias AS (SELECT CI.name, (COUNT(\*)/ (SELECT COUNT(\*) FROM PERFORMANCES)\*100) AS porcentaje\_icp FROM performer\_performances CI GROUP BY CI.name), resultado AS (SELECT PPG.name, PPG.porcentaje\_pp, PIP.porcentaje\_icp FROM porcentaje\_pistas\_grabadas PPG FULL OUTER JOIN porcentaje\_interpretaciones\_propias PIP ON PPG.name = PIP.name) SELECT DISTINCT name, porcentaje\_pp, porcentaje\_icp FROM resultado;

CREATE OR REPLACE PROCEDURE duracion\_media\_albums AS

duracion\_media\_c NUMBER;

duracion\_media\_v NUMBER;

duracion\_media\_m NUMBER;

duracion\_media\_s NUMBER;

BEGIN

SELECT AVG(CASE WHEN a.FORMAT = 'C' THEN t.DURATION \* t.TRACKS END) INTO duracion\_media\_c

FROM ALBUMS a

INNER JOIN (

SELECT PAIR, SUM(DURATION) AS DURATION, COUNT(\*) AS TRACKS

FROM TRACKS

GROUP BY PAIR

) t ON a.PAIR = t.PAIR

WHERE a.FORMAT = 'C' **AND WHERE ALBUMS.PERFORMER = curr\_user**;

SELECT AVG(CASE WHEN a.FORMAT = 'V' THEN t.DURATION \* t.TRACKS END) INTO duracion\_media\_v

FROM ALBUMS a

INNER JOIN (

SELECT PAIR, SUM(DURATION) AS DURATION, COUNT(\*) AS TRACKS

FROM TRACKS

GROUP BY PAIR

) t ON a.PAIR = t.PAIR

WHERE a.FORMAT = 'V' **AND WHERE ALBUMS.PERFORMER = curr\_user**;

SELECT AVG(CASE WHEN a.FORMAT = 'M' THEN t.DURATION \* t.TRACKS END) INTO duracion\_media\_m

FROM ALBUMS a

INNER JOIN (

SELECT PAIR, SUM(DURATION) AS DURATION, COUNT(\*) AS TRACKS

FROM TRACKS

GROUP BY PAIR

) t ON a.PAIR = t.PAIR

WHERE a.FORMAT = 'M' **AND WHERE ALBUMS.PERFORMER = curr\_user**;

SELECT AVG(CASE WHEN a.FORMAT = 'S' THEN t.DURATION \* t.TRACKS END) INTO duracion\_media\_s

FROM ALBUMS a

INNER JOIN (

SELECT PAIR, SUM(DURATION) AS DURATION, COUNT(\*) AS TRACKS

FROM TRACKS

GROUP BY PAIR

) t ON a.PAIR = t.PAIR

WHERE a.FORMAT = 'S' **AND WHERE ALBUMS.PERFORMER = curr\_user**;

DBMS\_OUTPUT.PUT\_LINE('La duración media de los álbumes de tipo C es: ' || duracion\_media\_c);

DBMS\_OUTPUT.PUT\_LINE('La duración media de los álbumes de tipo V es: ' || duracion\_media\_v);

DBMS\_OUTPUT.PUT\_LINE('La duración media de los álbumes de tipo M es: ' || duracion\_media\_m);

DBMS\_OUTPUT.PUT\_LINE('La duración media de los álbumes de tipo S es: ' || duracion\_media\_s);

END;

**--Cantidad media de canciones**

**SELECT AVG(NUM\_TRACKS) INTO num\_canciones\_conciertos FROM CONCERTS WHERE PERFORMER= curr\_user ;**

CREATE OR REPLACE PROCEDURE obtener\_estadisticas\_colaboradores AS

BEGIN

-- Obtener el número total de colaboraciones

SELECT

'Discográfica' AS tipo,

COUNT(\*) AS num\_colaboraciones

FROM albums

WHERE publisher IS NOT NULL

UNION ALL

SELECT

'Estudio' AS tipo,

COUNT(\*) AS num\_colaboraciones

FROM tracks

WHERE studio IS NOT NULL

UNION ALL

SELECT

'Ingeniero' AS tipo,

COUNT(\*) AS num\_colaboraciones

FROM tracks

WHERE engineer IS NOT NULL

UNION ALL

SELECT

'Manager - Álbumes' AS tipo,

COUNT(\*) AS num\_colaboraciones

FROM albums

WHERE manager IS NOT NULL

UNION ALL

SELECT

'Manager - Conciertos' AS tipo,

COUNT(\*) AS num\_colaboraciones

FROM concerts

WHERE manager IS NOT NULL;

-- Obtener las estadísticas de los colaboradores

SELECT

a.nombre,

a.tipo,

COUNT(DISTINCT t.pair) AS num\_colaboraciones\_album,

COUNT(DISTINCT c.performer) AS num\_colaboraciones\_concierto,

100.0 \* (COUNT(DISTINCT t.pair) + COUNT(DISTINCT c.performer)) /

(SELECT COUNT(\*) FROM (SELECT nombre FROM albums UNION SELECT manager FROM concerts) c) AS porcentaje\_total

FROM (

SELECT publisher AS nombre, 'Discográfica' AS tipo FROM albums WHERE publisher IS NOT NULL

UNION ALL

SELECT studio AS nombre, 'Estudio' AS tipo FROM tracks WHERE studio IS NOT NULL

UNION ALL

SELECT engineer AS nombre, 'Ingeniero' AS tipo FROM tracks WHERE engineer IS NOT NULL

UNION ALL

SELECT manager AS nombre, 'Manager' AS tipo FROM albums WHERE manager IS NOT NULL

UNION ALL

SELECT manager AS nombre, 'Manager' AS tipo FROM concerts WHERE manager IS NOT NULL

) a

LEFT JOIN albums alb ON alb.publisher = a.nombre OR alb.manager = a.nombre

LEFT JOIN tracks t ON t.pair = alb.pair

LEFT JOIN concerts c ON c.performer = a.nombre

GROUP BY a.nombre, a.tipo

UNION ALL

SELECT

nombre,

'Manager - Álbumes' AS tipo,

COUNT(DISTINCT alb.pair) AS num\_colaboraciones\_album,

0 AS num\_colaboraciones\_concierto,

100.0 \* COUNT(DISTINCT alb.pair) / (SELECT COUNT(\*) FROM albums WHERE manager = nombre) AS porcentaje\_total

FROM albums

WHERE manager IS NOT NULL

GROUP BY nombre

UNION ALL

SELECT

nombre,

'Manager - Conciertos' AS tipo,

0 AS num\_colaboraciones\_album,

COUNT(DISTINCT co.performer) AS num\_colaboraciones\_concierto,

100.0 \* COUNT(DISTINCT co.performer) / (SELECT COUNT(\*) FROM concerts WHERE manager = nombre) AS porcentaje\_total

FROM concerts co

WHERE manager IS NOT NULL

GROUP BY nombre;

END;

/

CREATE OR REPLACE PROCEDURE obtener\_estadisticas\_colaboradores AS

num\_total\_colaboraciones NUMBER;

BEGIN

-- Obtener el número total de colaboraciones

SELECT COUNT(\*) INTO num\_total\_colaboraciones

FROM (

SELECT publisher, null as manager, null as performer, null as studio, null as engineer FROM albums

UNION ALL

SELECT null as publisher, manager, null as performer, null as studio, null as engineer FROM albums

UNION ALL

SELECT null as publisher, null as manager, performer, null as studio, null as engineer FROM concerts

UNION ALL

SELECT null as publisher, null as manager, null as performer, studio, null as engineer FROM tracks

UNION ALL

SELECT null as publisher, null as manager, null as performer, null as studio, engineer FROM tracks

) colab;



-- Obtener las estadísticas de los colaboradores



SELECT

a.nombre,

a.tipo,

COUNT(DISTINCT t.pair) AS num\_colaboraciones\_album,

COUNT(DISTINCT c.performer) AS num\_colaboraciones\_concierto,

100.0 \* (COUNT(DISTINCT t.pair) + COUNT(DISTINCT c.performer)) / (

SELECT COUNT(DISTINCT t.pair) + COUNT(DISTINCT c.performer)

FROM tracks t

FULL OUTER JOIN concerts c

ON CAST(c.manager AS VARCHAR2(100)) = a.nombre

WHERE t.pair IS NOT NULL OR c.performer IS NOT NULL

) AS porcentaje\_total

FROM (

SELECT publisher AS nombre, 'Discográfica' AS tipo FROM albums WHERE publisher IS NOT NULL

UNION ALL

SELECT studio AS nombre, 'Estudio' AS tipo FROM tracks WHERE studio IS NOT NULL

UNION ALL

SELECT engineer AS nombre, 'Ingeniero' AS tipo FROM tracks WHERE engineer IS NOT NULL

UNION ALL

SELECT CAST(manager AS VARCHAR2(100)) AS nombre, 'Manager' AS tipo FROM albums WHERE manager IS NOT NULL

UNION ALL

SELECT CAST(manager AS VARCHAR2(100)) AS nombre, 'Manager' AS tipo FROM concerts WHERE manager IS NOT NULL

) a

LEFT JOIN albums alb ON alb.publisher = a.nombre OR alb.manager = a.nombre

LEFT JOIN tracks t ON t.studio = a.nombre OR t.engineer = a.nombre

LEFT JOIN concerts c ON c.performer = a.nombre

GROUP BY a.nombre, a.tipo

SELECT a.nombre, a.tipo, COUNT(DISTINCT t.pair) AS num\_colaboraciones\_album, COUNT(DISTINCT c.performer) AS num\_colaboraciones\_concierto, 100.0 \* (COUNT(DISTINCT t.pair) + COUNT(DISTINCT c.performer)) / (SELECT COUNT(DISTINCT t.pair) + COUNT(DISTINCT c.performer) AS num\_total\_colaboraciones FROM tracks t FULL OUTER JOIN concerts c ON CAST(c.manager AS VARCHAR2(100)) = a.nombre WHERE t.pair IS NOT NULL OR c.performer IS NOT NULL) AS porcentaje\_total

FROM (SELECT publisher AS nombre, 'Discográfica' AS tipo FROM albums WHERE publisher IS NOT NULL UNION ALL SELECT studio AS nombre, 'Estudio' AS tipo FROM tracks WHERE studio IS NOT NULL UNION ALL SELECT engineer AS nombre, 'Ingeniero' AS tipo FROM tracks WHERE engineer IS NOT NULL UNION ALL SELECT CAST(manager AS VARCHAR2(100)) AS nombre, 'Manager' AS tipo FROM albums WHERE manager IS NOT NULL UNION ALL SELECT CAST(manager AS VARCHAR2(100)) AS nombre, 'Manager' AS tipo FROM concerts WHERE manager IS NOT NULL) a

LEFT JOIN albums alb ON alb.publisher = a.nombre;

UNION ALL

SELECT

m.manager AS nombre,

'Manager - Álbumes' AS tipo,

COUNT(DISTINCT alb.pair) AS num\_colaboraciones\_album,

0 AS num\_colaboraciones\_concierto,

100.0 \* COUNT(DISTINCT alb.pair) / (SELECT COUNT(\*) FROM albums WHERE manager = m.manager) AS porcentaje\_total

FROM albums alb

INNER JOIN (

SELECT DISTINCT manager FROM albums

UNION

SELECT DISTINCT manager FROM concerts

) m ON alb.manager = m.manager

GROUP BY m.manager

UNION ALL

SELECT

m.manager AS nombre,

'Manager - Conciertos' AS tipo,

0 AS num\_colaboraciones\_album,

COUNT(DISTINCT co.performer) AS num\_colaboraciones\_concierto,

100.0 \* COUNT(DISTINCT co.performer) / (SELECT COUNT(\*) FROM concerts WHERE manager = m.manager) AS porcentaje\_total

FROM concerts co

INNER JOIN (

SELECT DISTINCT manager FROM albums

UNION

SELECT DISTINCT manager FROM concerts

) m ON co.manager = m.manager

GROUP BY m.manager;

END;

Las estadísticas de los colaboradores consisten en listar el nombre de cada uno de ellos y especificar su tipo (publisher, studio, engineer, manager) y el número de colaboraciones en álbumes/conciertos (en el caso de managers, listar por separado álbumes y conciertos) teniendo en cuenta que no existen las tablas colaboraciones\_album y colaboraciones\_concierto, que manager y discografica son columnas de la tabla album y que manager es una columna de concierto. Ademas, la tabla tracks se relaciona con albumes mediante el identificador pair. La tabla tracks tiene como columnas estudio e ingeniero y que la tabla nombre no existe en album

SELECT AVG(EXTRACT(YEAR FROM a.REL\_DATE) - EXTRACT(YEAR FROM b.max\_rel\_date))

INTO periodicidad\_media\_c

FROM ALBUMS a

JOIN (SELECT MAX(REL\_DATE) AS max\_rel\_date

FROM ALBUMS

WHERE PERFORMER = curr\_user AND FORMAT = 'C') b

ON b.max\_rel\_date < a.REL\_DATE

WHERE a.FORMAT = 'C' AND a.PERFORMER = curr\_user;

DISPARADORES:

(Pruebas)

**a) Cada vez que se inserte una interpretación de una canción en un concierto, se debe actualizar la duración del concierto sumando la duración de ese tema (opcionalmente, se pueden observar borrados y modificaciones de interpretaciones existentes).**

CREATE OR REPLACE TRIGGER update\_concert\_duration

AFTER INSERT OR UPDATE OR DELETE ON PERFORMANCES

FOR EACH ROW

DECLARE

total\_duration NUMBER(4);

BEGIN

SELECT SUM(duration) INTO total\_duration

FROM PERFORMANCES

WHERE performer = :NEW.performer AND when = :NEW.when;

UPDATE CONCERTS

SET duration = total\_duration

WHERE performer = :NEW.performer AND when = :NEW.when;

END;

**b) Rechazar compra de tickets por parte de clientes menores de 18 años.**

CREATE OR REPLACE TRIGGER reject\_underage\_purchase

BEFORE INSERT ON ATTENDANCES

FOR EACH ROW

DECLARE

client\_birthdate DATE;

client\_age NUMBER(3);

BEGIN

SELECT birthdate INTO client\_birthdate FROM CLIENTS WHERE e\_mail = :NEW.client;

client\_age := TRUNC(MONTHS\_BETWEEN(SYSDATE, client\_birthdate)/12);

IF client\_age < 18 THEN

RAISE\_APPLICATION\_ERROR(-20000, 'Purchase rejected: client is under 18 years old');

END IF;

END;

**c) Impedir que se inserte en ‘songs’ la misma canción con sus autores al revés.**

CREATE OR REPLACE TRIGGER reverse\_writers

BEFORE INSERT ON SONGS

FOR EACH ROW

DECLARE

reverse\_exists NUMBER;

BEGIN

SELECT COUNT(\*) INTO reverse\_exists

FROM SONGS

WHERE title = :NEW.title AND writer = :NEW.cowriter AND cowriter = :NEW.writer;

IF reverse\_exists > 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'La misma canción con los autores al revés ya existe en la tabla SONGS');

END IF;

END;

CREATE OR REPLACE PROCEDURE insertar\_album\_cancion(

p\_album\_nombre IN VARCHAR2,

p\_cancion\_nombre IN VARCHAR2,

p\_cancion\_writer IN VARCHAR2,

p\_cancion\_recdate IN DATE,

p\_cancion\_studio IN VARCHAR2,

p\_cancion\_engineer IN VARCHAR2,

p\_cancion\_duration IN NUMBER,

p\_cancion\_sequ IN NUMBER,

p\_pair IN VARCHAR2,

p\_format IN VARCHAR2,

p\_publisher IN VARCHAR2,

p\_manager IN VARCHAR2)

IS

v\_interprete\_actual VARCHAR2(50);

v\_album\_id VARCHAR2(50);

BEGIN

-- Verificar que todos los parámetros tengan un valor

IF p\_album\_nombre IS NULL OR p\_cancion\_nombre IS NULL OR p\_cancion\_writer IS NULL OR p\_cancion\_recdate IS NULL OR p\_cancion\_studio IS NULL OR p\_cancion\_engineer IS NULL OR p\_cancion\_duration IS NULL OR p\_cancion\_sequ IS NULL OR p\_pair IS NULL OR p\_format IS NULL OR p\_publisher IS NULL OR p\_manager IS NULL THEN

dbms\_output.put\_line('Error: se deben proporcionar valores para todos los parámetros.');

RETURN;

END IF;

SELECT USER INTO v\_interprete\_actual FROM dual;

-- Check if an album with the given pair already exists

SELECT pair INTO v\_album\_id FROM ALBUMS WHERE pair = p\_pair;

-- If no album with the given pair exists, insert a new one

IF v\_album\_id IS NULL THEN

INSERT INTO ALBUMS(title, performer, rel\_date, pair, format, publisher, manager)

VALUES(p\_album\_nombre, v\_interprete\_actual, TO\_CHAR(SYSDATE, 'YYYY'), p\_pair, p\_format, p\_publisher, p\_manager);

-- Retrieve the album ID of the newly inserted album

SELECT pair INTO v\_album\_id FROM ALBUMS WHERE title = p\_album\_nombre AND performer = v\_interprete\_actual AND pair = p\_pair;

END IF;

-- Insert the new track into the TRACKS table

INSERT INTO TRACKS(pair, title, duration, rec\_date, studio, engineer, sequ, writer)

VALUES(v\_album\_id, p\_cancion\_nombre, p\_cancion\_duration, p\_cancion\_recdate, p\_cancion\_studio, p\_cancion\_engineer, p\_cancion\_sequ, p\_cancion\_writer);

-- Commit the transaction

COMMIT;

-- Output a success message

dbms\_output.put\_line('Nuevo álbum y canción agregados exitosamente');

EXCEPTION

-- If there is an error, output the error message and roll back the transaction

WHEN OTHERS THEN

dbms\_output.put\_line('Error al agregar nuevo álbum y canción: ' || SQLERRM);

ROLLBACK;

END;