**Universidad Tecnológica de Panamá**

**Facultad de Ingeniería de Sistemas Computacionales**

**Licenciatura en Ingeniería de Sistemas de Información**

**Departamento de Sistemas de Información**

Sistemas de Bases de Datos II

Asignación #2 : cursores

Facilitador:

Ing. Henry Lezcano

Integrantes grupo 2:

Karen Cabrera 20-14-5403

Grupo:

1IF131

II Semestre, 2020

**UNIVERSIDAD TECNOLOGICA DE PANAMA**

**FACULTAD DE INGENIERIA DE SISTEMAS COMPUTACIONES**

**LICENCIATURA EN INGENIERIA DE SISTEMAS DE INFORMACION**

**SISTEMAS DE BASE DE DATOS II**

**Asignación #2**

1. Implemente el bloque pl/sql de la ppt No.4, donde en el área de ejecución del bloque, la información extraída por el cursor sea cargada en una relación o tabla diseñada por usted, establezca controles para el proceso la inserción de la información.

**--creando la tabla**

CREATE TABLE Students (

StudentId number(5) not null,

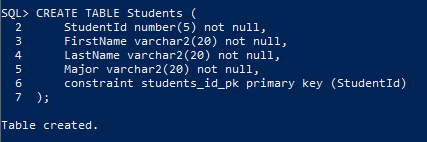
FirstName varchar2(20) not null,

LastName varchar2(20) not null,

Major varchar2(20) not null,

constraint students\_id\_pk primary key (StudentId)

);



**--insertando información en la tabla**

**INSERT INTO Students values (01, 'Andres', 'Diaz', 'Literatura');**

**INSERT INTO Students values (02, 'Bruno', 'Chee', 'Arquitectura');**

**INSERT INTO Students values (03, 'Viktor', 'Krum', 'Deportes');**

**INSERT INTO Students values (04, 'Mariana', 'Rueda', 'Medicina');**

**INSERT INTO Students values (05, 'Elsa', 'Padilla', 'Arte');**

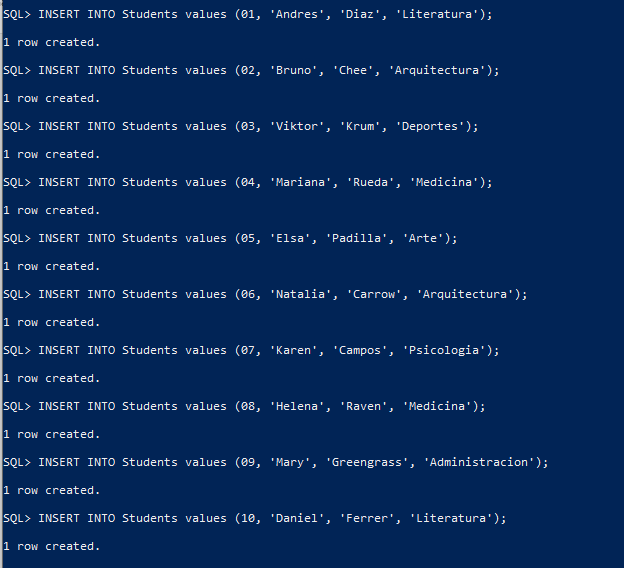
**INSERT INTO Students values (06, 'Natalia', 'Carrow', 'Arquitectura');**

**INSERT INTO Students values (07, 'Karen', 'Campos', 'Psicologia');**

**INSERT INTO Students values (08, 'Helena', 'Raven', 'Medicina');**

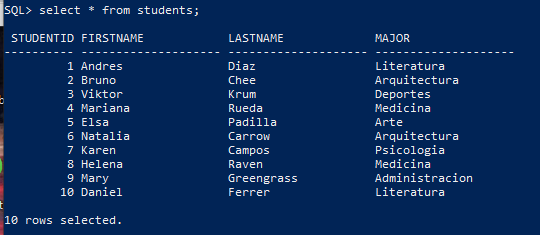
**INSERT INTO Students values (09, 'Mary', 'Greengrass', 'Administracion');**

**INSERT INTO Students values (10, 'Daniel', 'Ferrer', 'Literatura');**



**--consultando la tabla**

select \* from students;



**--eliminamos la tabla**

drop table students;



--**levantamos el server**

set serveroutput on;



**--realizamos el bloque**

DECLARE

v\_StudentID students.studentid%TYPE;

v\_FirstName students.firstname%TYPE;

v\_LastName students.lastname%TYPE;

/\* Valores de acoplamiento utilizado en la consulto \*/

v\_Major students.major%TYPE := '&Major';

/\* Declaración del Cursor \*/

CURSOR c\_Students IS

SELECT studentid, firstname, lastname

FROM students

WHERE major = v\_Major;

BEGIN

OPEN c\_Students;

LOOP

FETCH c\_Students INTO v\_StudentID, v\_FirstName, v\_LastName;

EXIT WHEN c\_Students%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Id: ' || v\_studentid || ' - Nombre: ' || v\_Firstname || ' ' || v\_LastName);

END LOOP;

EXCEPTION

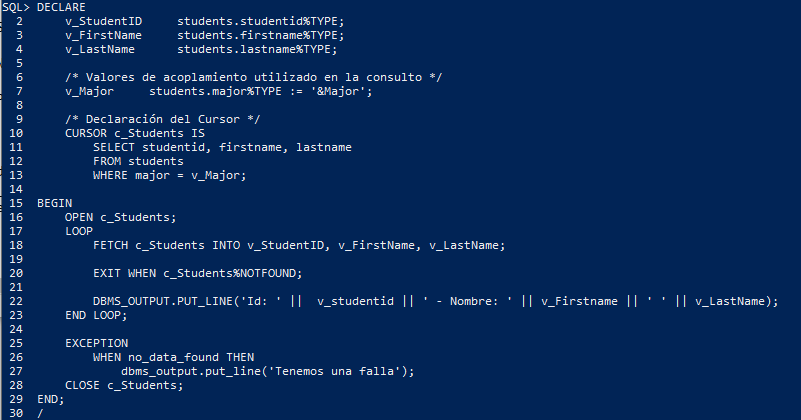
WHEN no\_data\_found THEN

dbms\_output.put\_line('Tenemos una falla');

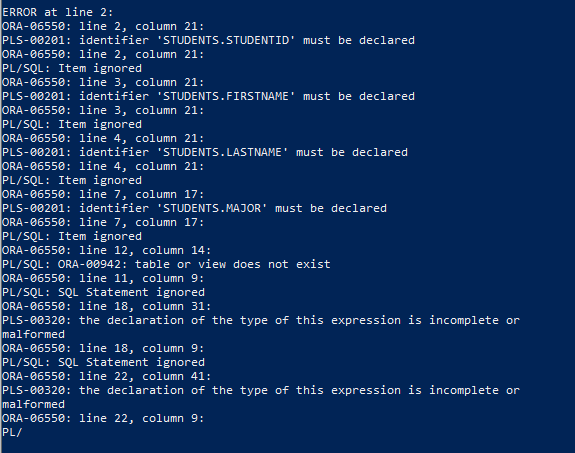
CLOSE c\_Students;

END;

/



Resultado



1. Complete el bloque pl/sql de la ppt No.8 de manera que podamos observar que información está extrayendo el cursor basado en variables de acoplamiento.

**--creando la tabla para los bloques**

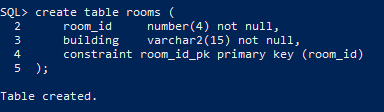
create table rooms (

room\_id number(4) not null,

building varchar2(15) not null,

constraint room\_id\_pk primary key (room\_id)

);



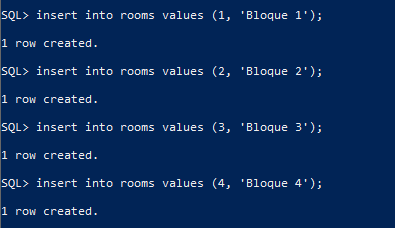
**---Insertando valores**

insert into rooms values (1, 'Bloque 1');

insert into rooms values (2, 'Bloque 2');

insert into rooms values (3, 'Bloque 3');

insert into rooms val ues (4, 'Bloque 4');



**--- creando la tabla para las clases**

create table classes (

class\_id number(5) not null,

department varchar2(30) not null,

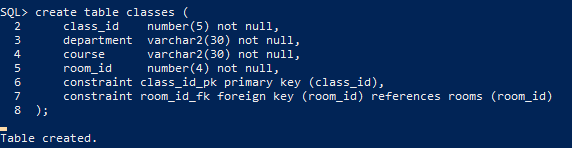
course varchar2(30) not null,

room\_id number(4) not null,

constraint class\_id\_pk primary key (class\_id),

constraint room\_id\_fk foreign key (room\_id) references rooms (room\_id)

);



**--Insertando información**

insert into classes values (101, 'Comunicacion', 'debate', 1);

insert into classes values (102, 'Comunicacion', 'Literatura', 1);

insert into classes values (201, 'Artes', 'acuarela', 2);

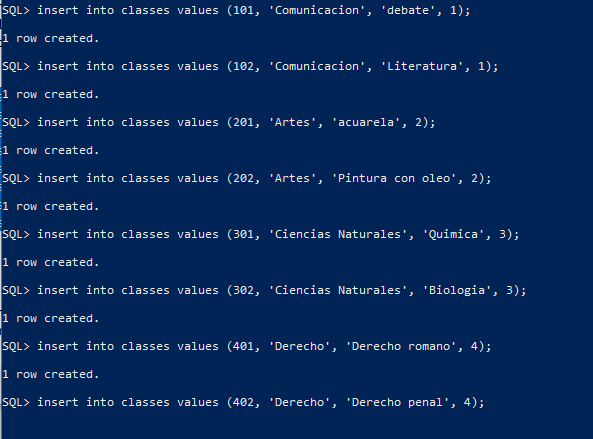
insert into classes values (202, 'Artes', 'Pintura con oleo', 2);

insert into classes values (301, 'Ciencias Naturales', 'Quimica', 3);

insert into classes values (302, 'Ciencias Naturales', 'Biologia', 3);

insert into classes values (401, 'Derecho', 'Derecho romano', 4);

insert into classes values (402, 'Derecho', 'Derecho penal', 4);



**--creando el bloque**

set serveroutput on;

DECLARE

v\_RoomID classes.room\_id%TYPE;

v\_Building rooms.building%TYPE;

v\_Department classes.department%TYPE;

v\_Course classes.course%TYPE;

CURSOR c\_Building IS

SELECT building

FROM rooms, classes

WHERE rooms.room\_id = classes.room\_id

AND department = v\_Department

AND course = v\_Course;

BEGIN

-- Asignar las variables de Acoplamiento antes de abrir el cursor

v\_Department := 'artes';

v\_Course := 'pintura con oleo';

OPEN c\_Building;

FETCH c\_Building INTO v\_building;

v\_Department := 'Ciencias naturales';

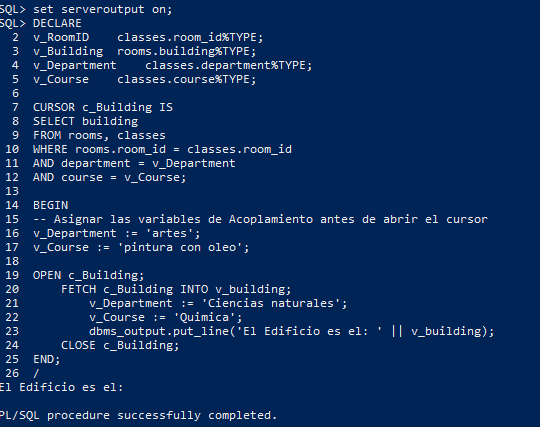
v\_Course := 'Quimica';

dbms\_output.put\_line('El Edificio es el: ' || v\_building);

CLOSE c\_Building;

END;

/



1. Modifique el bloque anterior aplicando el concepto de cursores parametrizados.

set serveroutput on;

DECLARE

v\_RoomID classes.room\_id%TYPE;

v\_Building rooms.building%TYPE;

v\_Department classes.department%TYPE;

v\_Course classes.course%TYPE;

CURSOR c\_Building (v\_Department classes.department%TYPE, v\_Course classes.course%TYPE) IS

SELECT building

FROM rooms, classes

WHERE rooms.room\_id = classes.room\_id

AND department = v\_Department

AND course = v\_Course;

BEGIN

-- Asignar las variables de Acoplamiento antes de abrir el cursor

v\_Department := 'comunicacion';

v\_Course := 'literatura';

OPEN c\_Building ('Artes', 'Acuarela');

FETCH c\_Building INTO v\_building;

v\_Department := 'Ciencias Naturales';

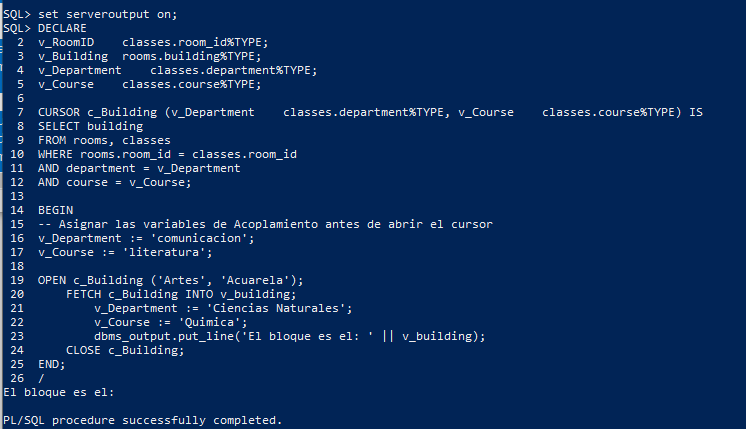
v\_Course := 'Quimica';

dbms\_output.put\_line('El bloque es el: ' || v\_building);

CLOSE c\_Building;

END;

/



1. 4. Implemente el bloque pl/sql de la ppt No.17 donde se valida el uso de los atributos para los cursores implícitos y la cláusula SELECT.

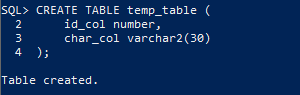
**--creando la tabla**

CREATE TABLE temp\_table (

id\_col number,

char\_col varchar2(30)

);



**--creando el bloque**

DECLARE

-- Registro para almacenar la información acerca de una clase.

v\_RoomData rooms%ROWTYPE;

BEGIN

-- Extraer la información sobre la clase ID -1

SELECT \* INTO v\_RoomData

FROM rooms

WHERE room\_id = -1;

/\* La siguiente orden no se ejecutará nunca,

ya que el control pasa inmediatamente al gestor de excepciones \*/

IF SQL%NOTFOUND THEN

INSERT INTO temp\_table ( char\_col) VALUES ( 'Not Found');

END IF;

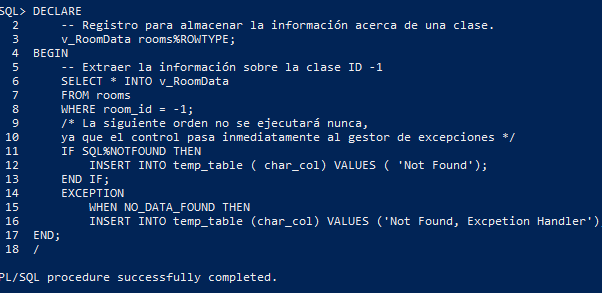
EXCEPTION

WHEN NO\_DATA\_FOUND THEN

INSERT INTO temp\_table (char\_col) VALUES ('Not Found, Excpetion Handler');

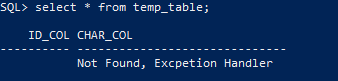
END;

/



**--haciendo una consulta**

select \* from temp\_table;



1. Implementos lo bloques pl/sql que se detalla en las ppt No.19, 20, 21, 22 que aplicación el ciclo de repetición integrado a los cursores basado reglas establecidas.

**--creamos la tabla de registro**

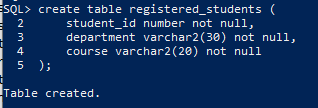
create table registered\_students (

student\_id number not null,

department varchar2(30) not null,

course varchar2(20) not null

);



**--declarando el bloque con su creación de cursor**

set serveroutput on;

DECLARE

/\* Declaración de variables para almacenar información acerca de los estudiantes

que cursan la especialidad de literatura \*/

v\_StudentID students.studentid%TYPE;

v\_FirstName students.firstname%TYPE;

v\_LastName students.lastname%TYPE;

-- Cursor para recuperar la informacion sobre los estudiantes de literatura

CURSOR c\_literatureStudents IS

SELECT studentid, firstname, lastname

FROM students

WHERE major = 'literatura';

BEGIN

--Abre el cursor e inicializa el conjunto activo

OPEN c\_literatureStudents;

LOOP

-- Recupera la información del siguiente estudiante

FETCH c\_literatureStudents INTO v\_StudentID, v\_FirstName, v\_LastName ;

--Salida del bucle cuando no hay más filas por recuperar

EXIT WHEN c\_literatureStudents%NOTFOUND ;

INSERT INTO registered\_students (student\_id, department, course)

VALUES ( v\_StudentID, 'Comunicacion', 'literatura');

INSERT INTO temp\_table ( id\_col, char\_col)

VALUES ( v\_studentID, v\_FirstName || ' '|| v\_LastName);

END LOOP;

-- Libera los recursos utilizados por el curso

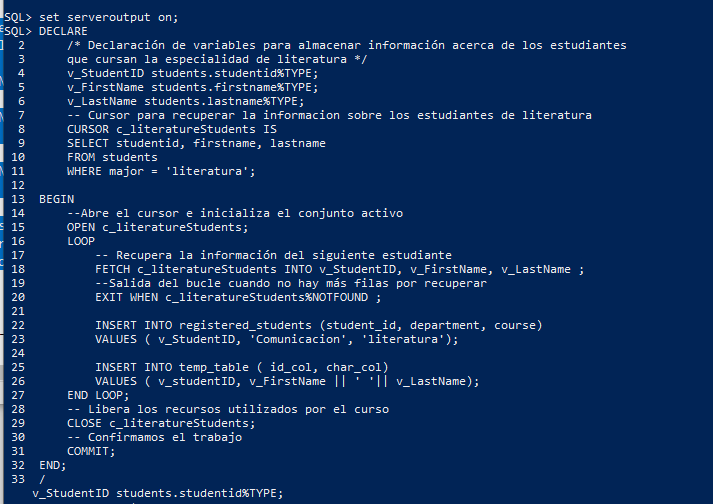
CLOSE c\_literatureStudents;

-- Confirmamos el trabajo

COMMIT;

END;

/



**--segundo bloque**

DECLARE

/\* Declaración de variables para almacenar información acerca de los estudiantes

que cursan la especialidad de literatura \*/

v\_StudentID students.studentid%TYPE;

v\_FirstName students.firstname%TYPE;

v\_LastName students.lastname%TYPE;

-- Cursor para recuperar la información sobre los estudiantes de literatura

CURSOR c\_literatureStudents IS

SELECT studentid, firstname, lastname

FROM students

WHERE major = 'literatura';

BEGIN

-- Abre el cursor e inicializa el conjunto activo

OPEN c\_literatureStudents;

LOOP

-- Recupera la información del siguiente estudiante

FETCH c\_literatureStudents INTO v\_StudentID, v\_FirstName, v\_LastName ;

INSERT INTO registered\_students (student\_id, department, course)

VALUES ( v\_StudentID, 'comunicacion', 'literatura');

INSERT INTO temp\_table (id\_col, char\_col)

VALUES ( v\_studentID, v\_FirstName || ' '|| v\_LastName);

-- Salida del bucle cuando no hay más filas por recuperar

EXIT WHEN c\_literatureStudents%NOTFOUND ;

END LOOP;

-- Libera los recursos utilizados por el curso

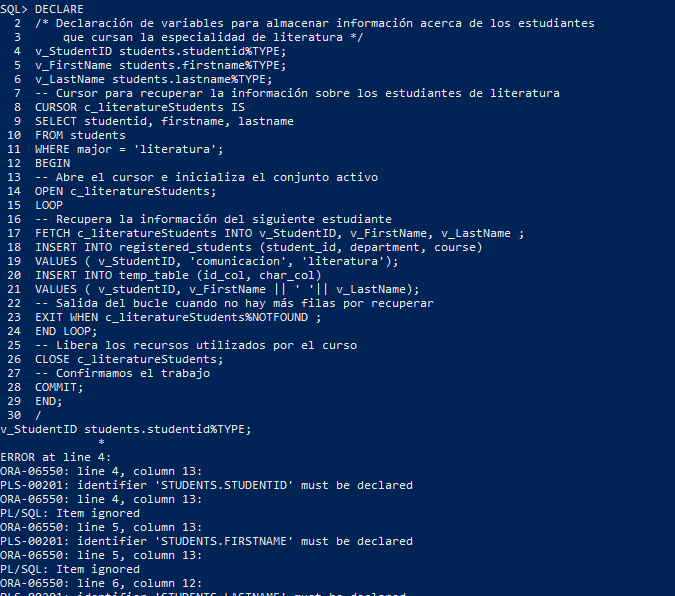
CLOSE c\_literatureStudents;

-- Confirmamos el trabajo

COMMIT;

END;

/



**---tercer bloque**

DECLARE

-- Cursor para recuperar la información sobre los estudiantes de literatura

CURSOR c\_literatureStudents IS

SELECT studentid, firstname, lastname

FROM students

WHERE major = 'literatura';

-- Declaración el registro para almacenar información extraída

v\_StudentData c\_literatureStudents%ROWTYPE;

BEGIN

-- Abre el cursor e inicializa el conjunto activo

OPEN c\_literatureStudents;

-- Recupera la información del siguiente estudiante

FETCH c\_literatureStudents INTO v\_StudentData;

-- El bucle continua mientras haya mas filas que extraer

WHILE c\_literatureStudents%FOUND LOOP

INSERT INTO registered\_students( student\_id, department, course)

VALUES ( v\_StudentData.studentid, 'comunicacion', 'literatura');

INSERT INTO temp\_table (id\_col, char\_col)

VALUES ( v\_StudentData.studentid, v\_StudentData.firstname || ' '|| v\_StudentData.lastname);

-- Recuperar la fila siguiente. La condición %FOUND se comprobara antes de que el bucle continúe

FETCH c\_HistoryStudents INTO v\_StudentData;

END LOOP;

-- Libera los recursos utilizados por el curso

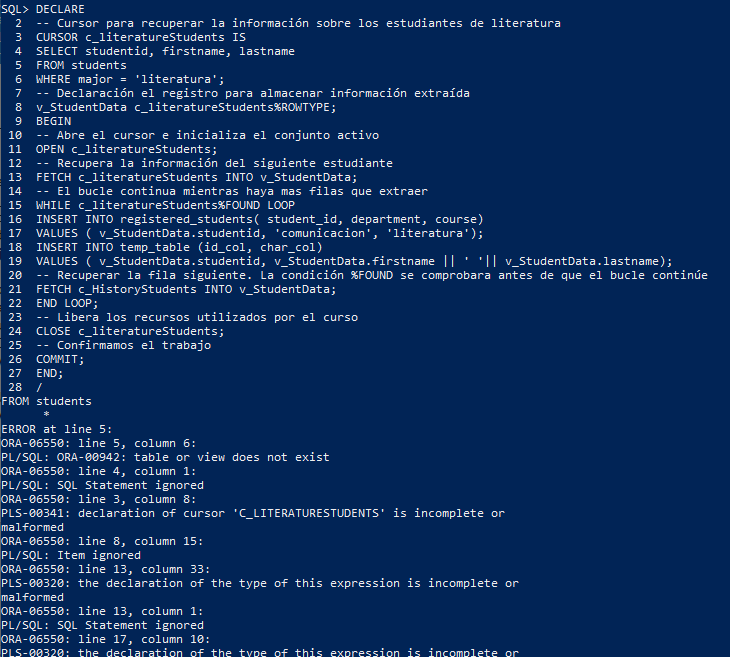
CLOSE c\_literatureStudents;

-- Confirmamos el trabajo

COMMIT;

END;

/



**--cuarto bloque—**

DECLARE

-- Cursor para recuperar la información sobre los estudiantes de literature

CURSOR c\_literatureStudents IS

SELECT studentid, firstname, lastname

FROM students

WHERE major = 'literatura';

BEGIN

/\* Inicio del bucle. Aquí se ejecuta una orden OPEN

implícita sobre c\_literatureStudents \*/

FOR v\_StudentData IN c\_literatureStudents LOOP

INSERT INTO registered\_students (student\_id, department, course)

VALUES (v\_StudentData.studentid, 'Comunicacion', 'literatura');

INSERT INTO temp\_table (id\_col, char\_col)

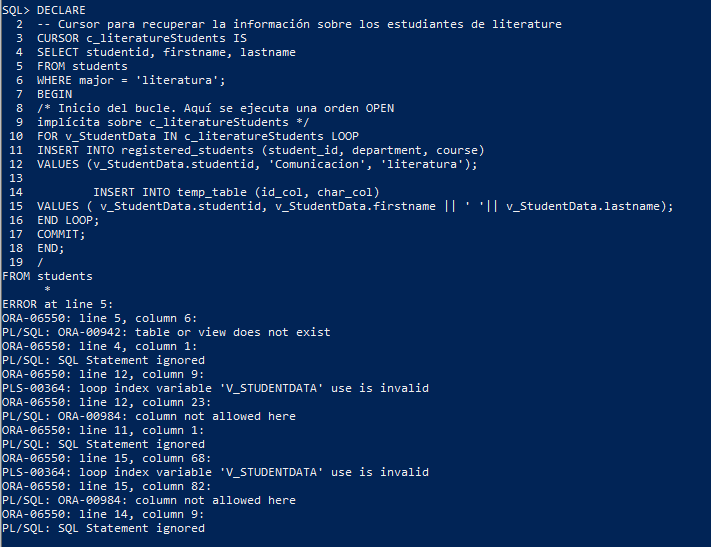
VALUES ( v\_StudentData.studentid, v\_StudentData.firstname || ' '|| v\_StudentData.lastname);

END LOOP;

COMMIT;

END;

/



1. Diseñe bloque pl/sql que extraiga información de una relación o tabla de base de datos por medio de un cursor y esta se actualizada en el área de ejecución del proceso estableciendo los controles para el proceso de actualización.

**---creando la tabla películas**

create table peliculas (

id\_pelicula number not null,

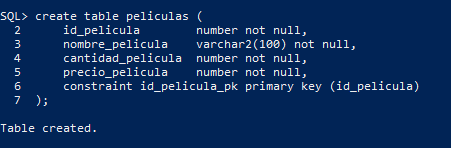
nombre\_pelicula varchar2(100) not null,

cantidad\_pelicula number not null,

precio\_pelicula number not null,

constraint id\_pelicula\_pk primary key (id\_pelicula)

);



**---insertando valores**

insert into peliculas values (1, 'atrapame si puedes', 1, 40);

insert into peliculas values (2, 'los huegos del hambre', 3, 60);

insert into peliculas values (3, 'star wars', 9, 90);

insert into peliculas values (4, 'harry potter', 8, 60);

insert into peliculas values (5, 'rapidos y furiosos', 10, 50);

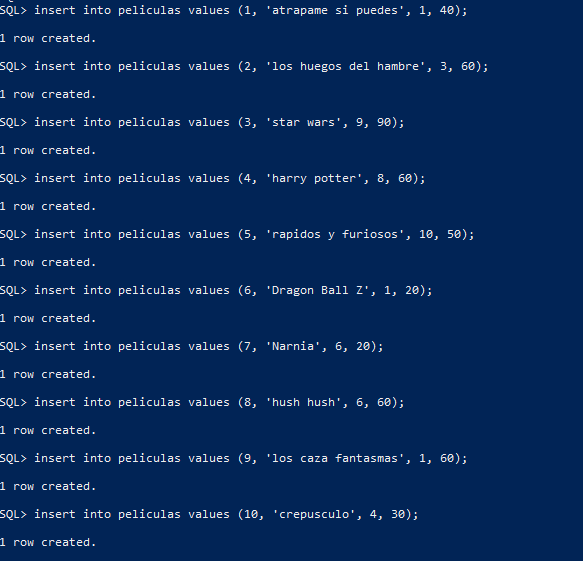
insert into peliculas values (6, 'Dragon Ball Z', 1, 20);

insert into peliculas values (7, 'Narnia', 6, 20);

insert into peliculas values (8, 'hush hush', 6, 60);

insert into peliculas values (9, 'los caza fantasmas', 1, 60);

insert into peliculas values (10, 'crepusculo', 4, 30);



**--creando el bloque**

DECLARE

v\_nombre\_pelicula pelicula.nombre\_pelicula%TYPE;

v\_precio\_pelicula pelicula.precio\_pelicula%TYPE;

v\_descuento number := 5;

CURSOR c\_descuento IS

SELECT nombre\_pelicula, precio\_pelicula

FROM peliculas

WHERE cantidad\_pelicula >= v\_descuento;

BEGIN

OPEN c\_descuento;

UPDATE peliculas

set precio\_pelicula = precio\_pelicula - (precio\_pelicula\*0.5)

where cantidad\_pelicula >= 5;

dbms\_output.put\_line('peliculas con descuento');

LOOP

FETCH c\_descuento INTO v\_nombre\_pelicula, v\_precio\_pelicula;

EXIT WHEN c\_descuento%NOTFOUND;

dbms\_output.put\_line(v\_nombre\_pelicula);

END LOOP;

CLOSE c\_descuento;

COMMIT;

END;

/

