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.

```
SQL> --creacioon de tabla students
SQL> CREATE TABLE students (
  2  id varchar2(15) not null,
  3  first_name varchar2(20) not null,
  4  last_name varchar2(20) not null,
  5  major varchar2(30) not null,
  6  primary key (id)
  7  );

Table created.

SQL> COMMIT;

Commit complete.

SQL>
SQL> CREATE TABLE estudiante_select (
  2  id varchar2(15) not null,
  3  first_name varchar2(20) not null,
  4  last_name varchar2(20) not null,
  5  primary key (id)
  6  );

Table created.

SQL> COMMIT;

Commit complete.
```

```
SQL> --insercion de datos
SQL> Insert into students values ( '9-433-222', 'Laura', 'Fernandez', 'Art History ');

1 row created.

SQL> Insert into students values ( '8-776-133', 'Maximina', 'Mendez', 'Computer Science');

1 row created.

SQL> Insert into students values ( '4-297-200', 'Alberto', 'Urieta', 'Computer Science');

1 row created.

SQL> Insert into students values ( '9-433-2182', 'Juan', 'Fernandez', 'Art History ');

1 row created.

SQL> Insert into students values ( '7-766-130', 'Marcos', 'Carrizo', 'Computer Science');

1 row created.

SQL> Insert into students values ( '5-298-212', 'Adalberto', 'Mendez', 'Developer');
```

```
SQL> set serveroutput on;
SQL> DECLARE
  2  --Variable de salida para almacenar los resultados de ls consulta
  3  v_StudentID students.id%TYPE;
  4  v_FirstName students.first_name%TYPE;
  5  v_LastName students.last_name%TYPE;
  6
  7  --valores de acoplamiento usados en la consulta
  8  v_major students.major%TYPE := 'Computer Science';
  9
 10  --declaracion del cursor
 11  CURSOR c_Students IS
 12  SELECT id, first_name, last_name
 13  FROM students
 14  WHERE major = v_major;
 15  BEGIN
 16  --identificar las filas en el conjunto activo y preparar el procesamiento de datos
 17  OPEN c_Students;
 18  LOOP
 19  --recuperar cada fila del conjutno active y almacenarlo en variables
 20  FETCH c_Students INTO v_StudentID, v_FirstName, v_LastName;
 21  --salir cuando no hayan filas
 22  EXIT WHEN c_Students%NOTFOUND;
 23  insert into estudiante_select values (v_StudentID, v_FirstName, v_LastName );
 24  DBMS_OUTPUT.put_line('ID:   '||v_StudentID||'      NAME:   ' || v_FirstName ||'    '||v_LastName);
 25
 26  END LOOP;
 27  CLOSE  c_Students;
 28  END;
 29  /
ID:   8-776-133      NAME:   Maximina    Mendez
ID:   4-297-200      NAME:   Alberto    Urieta
ID:   7-766-130      NAME:   Marcos    Carrizo
```

2. 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.

```
SQL> CREATE TABLE rooms(
  2  room_id varchar2(15) not null,
  3  building varchar2(20) not null,
  4  primary key (room_id)
  5  );

Table created.

SQL> COMMIT;

Commit complete.

SQL> CREATE TABLE classes (
  2  course_id varchar2(15) not null primary key,
  3  department varchar2(20) not null,
  4  course varchar2(20) not null,
  5  room_id varchar2(15),
  6  constraint room_id_fk foreign key (room_id) references rooms (room_id)
  7  );

Table created.

SQL> COMMIT;

Commit complete.
```

```
SQL> Insert into rooms values (  '1-116', 'Edif. 1' );

1 row created.

SQL> Insert into rooms values (  '1-117', 'Edif. 1' );

1 row created.

SQL> Insert into rooms values (  '1-118', 'Edif. 1' );

1 row created.

SQL> Insert into rooms values (  '3-319', 'Edif. 3' );

1 row created.

SQL> Insert into rooms values (  '1-120', 'Edif. 1' );

1 row created.

SQL> Insert into rooms values (  '1-121', 'Edif. 1' );

1 row created.

SQL>
```

```
SQL> Insert into classes values ( 'HIS432', 'HIS', '101', '1-116');

1 row created.

SQL> Insert into classes values ( 'MAT321', 'MAT', '102','1-117');

1 row created.

SQL> Insert into classes values ( 'HIS433', 'HIS', '101','1-118');

1 row created.

SQL> Insert into classes values ( 'ENG786', 'ENG', '103','3-319');

1 row created.

SQL> Insert into classes values ( 'HIS434', 'HIS', '110', '1-120');

1 row created.

SQL> Insert into classes values ( 'HIS435' , 'HIS', '101', '1-121');

1 row created.
```

```
SQL> set serveroutput on;
SQL> DECLARE
  2   v_RoomID classes.room_id%TYPE;
  3  v_Building rooms.building%TYPE;
  4   v_Department classes.department%TYPE;
  5   v_Course classes.course%TYPE;
  6
  7   CURSOR c_Building IS
  8   SELECT building
  9   FROM rooms, classes
 10   WHERE rooms.room_id = classes.room_id
 11   AND department = v_Department
 12   AND course = v_Course;
 13  BEGIN
 14   -- Asignar las variables de Acoplamiento antes de abril el cursor
 15   v_Department := 'HIS';
 16   v_Course := 101;
 17          -- Abril el Cursor
 18  OPEN c_Building;
 19   -- Reasignar las variables de acoplamiento - No tienen efecto alguno, ya que el cursor esta abierto
 20       LOOP
 21  --recuperar cada fila del conjutno active y almacenarlo en variables
 22  FETCH c_Building INTO v_Building;
 23  --salir cuando no hayan filas
 24  EXIT WHEN c_Building%NOTFOUND;
 25  DBMS_OUTPUT.put_line('Curso dictado en:     '||  v_Building);
 26
 27        END LOOP;
 28   v_Department := 'XXX';
 29   v_Course := -1;
 30  END;
 31  /
Curso dictado en:     Edif. 1
Curso dictado en:     Edif. 1
Curso dictado en:     Edif. 1

PL/SQL procedure successfully completed.
```

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

```
SQL> set serveroutput on;
SQL> DECLARE
  2   v_RoomID classes.room_id%TYPE;
  3  v_Building rooms.building%TYPE;
  4
  5   CURSOR c_Building( v_Department classes.department%TYPE,
  6   v_Course classes.course%TYPE) IS
  7   SELECT building
  8   FROM rooms, classes
  9   WHERE rooms.room_id = classes.room_id
 10   AND department = v_Department
 11   AND course = v_Course;
 12  BEGIN
 13   -- Abril el Cursor
 14   OPEN c_Building ('HIS','101');
 15   -- Reasignar las variables de acoplamiento - No tienen efecto alguno, ya que el cursor esta abierto
 16  LOOP
 17  --recuperar cada fila del conjutno active y almacenarlo en variables
 18  FETCH c_Building INTO v_Building;
 19  --salir cuando no hayan filas
 20  EXIT WHEN c_Building%NOTFOUND;
 21  DBMS_OUTPUT.put_line('Curso dictado en:     '||  v_Building);
 22
 23  END LOOP;
 24
 25  END;
 26  /
Curso dictado en:     Edif. 1
Curso dictado en:     Edif. 1
Curso dictado en:     Edif. 1

PL/SQL procedure successfully completed.
```

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.

```
SQL> create table temp_table (
  2  char_col varchar2(50),
  3  num_col varchar2(50)
  4  );

Table created.

SQL> ---------------------------------------------------------------------------------------------------------------------
SQL> set serveroutput on;
SQL> DECLARE
  2  v_RoomData rooms%ROWTYPE;
  3  BEGIN
  4   -- Extraer la información sobre la clase ID -1
  5   SELECT * INTO v_RoomData
  6   FROM rooms
  7   WHERE room_id = '-1';
  8   /* La siguiente orden no se ejecutará nunca, ya que el control pasa inmediatamente al gestor de excepciones */
  9   IF SQL%NOTFOUND THEN
 10   INSERT INTO temp_table ( char_col) VALUES ( 'Not Found');
 11   END IF;
 12  EXCEPTION
 13   WHEN NO_DATA_FOUND THEN
 14   INSERT INTO temp_table ( char_col) VALUES ( 'Not Found, Excpetion Handler');
 15  END;
 16  /
```

```
SQL> select * from temp_table;

CHAR_COL
--------------------------------------------------
NUM_COL
--------------------------------------------------
Not Found, Excpetion Handler


Maximina Mendez
8-776-133

Alberto Urieta
4-297-200


CHAR_COL
--------------------------------------------------
NUM_COL
--------------------------------------------------
Marcos Carrizo
7-766-130

Maximina Mendez
8-776-133

Alberto Urieta
4-297-200
```

```
CHAR_COL
--------------------------------------------------
NUM_COL
--------------------------------------------------
Marcos Carrizo
7-766-130

Maximina Mendez
8-776-133

Alberto Urieta
4-297-200


CHAR_COL
--------------------------------------------------
NUM_COL
--------------------------------------------------
Marcos Carrizo
7-766-130

Marcos Carrizo
7-766-130

Maximina Mendez
8-776-133
```

```
CHAR_COL
--------------------------------------------------
NUM_COL
--------------------------------------------------
Alberto Urieta
4-297-200

Marcos Carrizo
7-766-130

Maximina Mendez
8-776-133


CHAR_COL
--------------------------------------------------
NUM_COL
--------------------------------------------------
Alberto Urieta
4-297-200

Marcos Carrizo
7-766-130

Not Found, Excpetion Handler
```

5. 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.

5.1

```
SQL> Create table registered_students(
  2  department varchar2(20) not null,
  3  course varchar2(20) not null,
  4  students_id varchar2(15),
  5  constraint students_id_fk foreign key (students_id ) references students (id ),
  6  course_id varchar2(15),
  7  constraint course_id_fk foreign key (course_id) references classes (course_id)
  8  );

Table created.
```

```
SQL> DECLARE
  2   /* Declaración de variables para almacenar información acerca de los estudiantes que cursan la especialidad de Historia */
  3   v_StudentID students.id%TYPE;
  4   v_FirstName students.first_name%TYPE;
  5   v_LastName students.last_name%TYPE;
  6  -- Cursor para recuperar la informacion sobre los estudiantes de Historia
  7   CURSOR c_HistoryStudents IS
  8   SELECT id, first_name, last_name
  9   FROM students
 10   WHERE major = 'Computer Science';
 11  BEGIN
 12   -- Abre el cursor e inicializa el conjunto activo
 13   OPEN c_HistoryStudents;
 14   LOOP
 15   -- Recupera la información del siguiente estudiante
 16   FETCH c_HistoryStudents INTO v_StudentID, v_FirstName, v_LastName ;
 17   -- Salida del bucle cuando no hay más filas por recuperar
 18   EXIT WHEN c_HistoryStudents%NOTFOUND ;
 19   /* Procesa las filas recuperadas. En este caso matricula a cada estudiante en Historia 301, insertándolo en la tabla registered_students.
 20  Registra también el nombre y el apellido en la tabla temp_table */
 21   INSERT INTO registered_students ( students_id, department, course)
 22   VALUES ( v_StudentID, 'HIS', '101');
 23   INSERT INTO temp_table ( num_col, char_col)
 24   VALUES ( v_studentID, v_FirstName || ' '|| v_LastName);
 25   END LOOP;
 26   -- Libera los recursos utilizados por el curso
 27   CLOSE c_HistoryStudents;
 28  -- Confirmamos el trabajo
 29  COMMIT;
 30  END;
 31  /
```

```
SQL> select * from registered_students;

DEPARTMENT           COURSE               STUDENTS_ID     COURSE_ID
-------------------- -------------------- --------------- ---------------
HIS                  101                  8-776-133
HIS                  101                  4-297-200
HIS                  101                  7-766-130
HIS                  101                  8-776-133
HIS                  101                  4-297-200
HIS                  101                  7-766-130

6 rows selected.
```

5.2

```
SQL> DECLARE
  2   /* Declaración de variables para almacenar información acerca de los estudiantes que cursan la especialidad de Historia */
  3   v_StudentID students.id%TYPE;
  4   v_FirstName students.first_name%TYPE;
  5   v_LastName students.last_name%TYPE;
  6  -- Cursor para recuperar la información sobre los estudiantes de Historia
  7   CURSOR c_HistoryStudents IS
  8   SELECT id, first_name, last_name
  9   FROM students
 10   WHERE major = 'Computer Science';
 11  BEGIN
 12   -- Abre el cursor e inicializa el conjunto activo
 13   OPEN c_HistoryStudents;
 14   LOOP
 15   -- Recupera la información del siguiente estudiante
 16   FETCH c_HistoryStudents INTO v_StudentID, v_FirstName, v_LastName ;
 17   /* Procesa las filas recuperadas. En este caso matricula a cada estudiante en Historia 301, insertándolo en la tabla registered_students.
 18  Registra también el nombre y el apellido en la tabla temp_table */
 19   INSERT INTO registered_students ( students_id, department, course)
 20   VALUES ( v_StudentID, 'HIS', 101);
 21   INSERT INTO temp_table ( num_col, char_col)
 22   VALUES ( v_studentID, v_FirstName || ' '|| v_LastName);
 23   -- Salida del bucle cuando no hay más filas por recuperar
 24   EXIT WHEN c_HistoryStudents%NOTFOUND ;
 25   END LOOP;
 26   -- Libera los recursos utilizados por el curso
 27   CLOSE c_HistoryStudents;
 28  -- Confirmamos el trabajo
 29  COMMIT;
 30  END;
 31  /
```

```
DEPARTMENT           COURSE               STUDENTS_ID     COURSE_ID
-------------------- -------------------- --------------- ---------------
HIS                  101                  8-776-133
HIS                  101                  4-297-200
HIS                  101                  7-766-130
HIS                  101                  8-776-133
HIS                  101                  4-297-200
HIS                  101                  7-766-130
HIS                  101                  8-776-133
HIS                  101                  4-297-200
HIS                  101                  7-766-130
HIS                  101                  7-766-130

10 rows selected.
```

5.3

```
SQL> DECLARE
  2   -- Cursor para recuperar la información sobre los estudiantes de Historia
  3   CURSOR c_HistoryStudents IS
  4   SELECT id, first_name, last_name
  5   FROM students
  6   WHERE major = 'Computer Science';
  7   -- Declaración el registro para almacenar información extraída
  8   v_StudentData c_HistoryStudents%ROWTYPE;
  9  BEGIN
 10   -- Abre el cursor e inicializa el conjunto activo
 11   OPEN c_HistoryStudents;
 12   -- Recupera la información del siguiente estudiante
 13   FETCH c_HistoryStudents INTO v_StudentData;
 14   -- El bucle continua mientras haya mas filas que extraer
 15   WHILE c_HistoryStudents%FOUND LOOP
 16   /* Procesa las filas recuperadas. En este caso matricula a cada estudiante en Historia 301, insertándolo en la tabla
 17  registered_students. Registra también el nombre y el apellido en la tabla temp_table */
 18   INSERT INTO registered_students( students_id, department, course)
 19   VALUES ( v_StudentData.ID, 'HIS', 101);
 20   INSERT INTO temp_table ( num_col, char_col)
 21   VALUES ( v_StudentData.ID, v_StudentData.first_name || ' '|| v_StudentData.last_name);
 22   -- Recuperar la fila siguiente. La condición %FOUND se comprobara antes de que el bucle continúe
 23   FETCH c_HistoryStudents INTO v_StudentData;
 24   END LOOP;
 25   -- Libera los recursos utilizados por el curso
 26   CLOSE c_HistoryStudents;
 27  -- Confirmamos el trabajo
 28  COMMIT;
 29  END;
 30  /
```

5.4

```
SQL> DECLARE
  2   -- Cursor para recuperar la información sobre los estudiantes de Historia
  3   CURSOR c_HistoryStudents IS
  4   SELECT id, first_name, last_name
  5   FROM students
  6   WHERE major = 'Computer Science';
  7  BEGIN
  8   /* Inicio del bucle. Aquí se ejecuta una orden OPEN
  9   implícita sobre c_HistoryStudents */
 10   FOR v_StudentData IN c_HistoryStudents LOOP
 11   -- Aquí se ejecuta una orden FETCH implícita
 12   /* Procesa las filas recuperadas. En este caso matricula a cada estudiante en Historia 301, insertándolo en la tabla
 13  registered_students. Registra también el nombre y el apellido en la tabla temp_table */
 14   INSERT INTO registered_students( students_id, department, course)
 15   VALUES ( v_StudentData.ID, 'HIS',101);
 16   INSERT INTO temp_table ( num_col, char_col)
 17   VALUES ( v_StudentData.ID, v_StudentData.first_name || ' '|| v_StudentData.last_name);
 18   -- Antes de continuar con el bucle, aquí se hace una comprobacion implícita de c_HistoryStudents %NOTFOUND.
 19   END LOOP;
 20  -- Ahora el bucle ha terminado, se hace cierre implícito del cursor c_HistoryStudents
 21  -- Confirmamos el trabajo
 22  COMMIT;
 23  END;
 24  /

PL/SQL procedure successfully completed.
```

```
DEPARTMENT           COURSE               STUDENTS_ID     COURSE_ID
-------------------- -------------------- --------------- ---------------
HIS                  101                  8-776-133
HIS                  101                  4-297-200
HIS                  101                  7-766-130
HIS                  101                  8-776-133
HIS                  101                  4-297-200
HIS                  101                  7-766-130
HIS                  101                  8-776-133
HIS                  101                  4-297-200
HIS                  101                  7-766-130
HIS                  101                  7-766-130
HIS                  101                  8-776-133

DEPARTMENT           COURSE               STUDENTS_ID     COURSE_ID
-------------------- -------------------- --------------- ---------------
HIS                  101                  4-297-200
HIS                  101                  7-766-130
HIS                  101                  8-776-133
HIS                  101                  4-297-200
HIS                  101                  7-766-130

16 rows selected.

SQL>
```

6.

```
SQL> create table Celulares(
  2      id_celular       number not null,
  3      nombre_celular    varchar2(100) not null,
  4      cantidad_celular  number not null,
  5      precio_celular  number not null,
  6      constraint id_juego_pk primary key (id_celular)
  7  );

Table created.
```

```
SQL>
SQL> insert into Celulares values (1, 'Samsung Galaxy A2 Core', 20, 700);

1 row created.

SQL> insert into Celulares values (2, 'Samsung Galaxy A10', 25, 800);

1 row created.

SQL> insert into Celulares values (3, 'Samsung Galaxy A20', 30, 900);

1 row created.

SQL> insert into Celulares values (4, 'Huawei P Smart 2019', 35,1000);

1 row created.

SQL> insert into Celulares values (5, 'Huawei Nova 3i', 40, 1100);

1 row created.

SQL> insert into Celulares values (6, 'Huawei Mate 20 RS Porsche Design', 45, 1200);

1 row created.

SQL> insert into Celulares values (7, 'iPhone 5', 50, 1400);

1 row created.

SQL> insert into Celulares values (8, 'Xiaomi Redmi K30', 55, 1300);
```

```
SQL> set serveroutput on;
SQL> DECLARE
  2      v_nombre_celular  Celulares.nombre_celular%TYPE;
  3      v_precio_celular  Celulares.precio_celular%TYPE;
  4
  5      v_descuento number := 10;
  6
  7      CURSOR c_descuento IS
  8          SELECT nombre_celular, precio_celular
  9          FROM Celulares
 10          WHERE cantidad_celular >= v_descuento;
 11
 12  BEGIN
 13      OPEN c_descuento;
 14      UPDATE Celulares
 15              set precio_celular = precio_celular - (precio_celular*0.5)
 16              where cantidad_celular >= 10;
 17      dbms_output.put_line('Juegos con descuento');
 18      LOOP
 19          FETCH c_descuento INTO v_nombre_celular, v_precio_celular;
 20          EXIT WHEN c_descuento%NOTFOUND;
 21
 22          dbms_output.put_line(v_nombre_celular);
 23      END LOOP;
 24      CLOSE c_descuento;
 25      COMMIT;
 26  END;
 27  /
```

```
Juegos con descuento
Samsung Galaxy A2 Core
Samsung Galaxy A10
Samsung Galaxy A20
Huawei P Smart 2019
Huawei Nova 3i
Huawei Mate 20 RS Porsche Design
iPhone 5
Xiaomi Redmi K30
```