

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS



“LABORATORIO 04”

ASIGNATURA

Administración de Base de Datos

DOCENTE

Chávez Soto, Jorge Luis

ESTUDIANTE

Carhuaricra Anco, Heidy Nicole - 23200150

Lima, Perú

2025

ÍNDICE

LABORATORIO 04.....	3
I. OBJETIVOS	3
II. RESUMEN	3
III. DESCRIPCIÓN DE LAS TABLAS.....	3
1. S (Suppliers).....	3
2. P (Partes).....	3
3. SP (Envíos)	4
4. J (Proyectos).....	4
5. SPJ (Envíos a Proyectos)	5
IV. EJERCICIOS DE PROCEDIMIENTOS Y FUNCIONES.....	6

LABORATORIO 04

I. OBJETIVOS

El presente laboratorio tiene por objetivos:

- Escribir procedimientos almacenados.
- Escribir funciones almacenadas.

II. RESUMEN

Dada la base de datos presentada a continuación, desarrolle los siguientes ejercicios.

III. DESCRIPCIÓN DE LAS TABLAS

1. S (Suppliers)

Representa a los proveedores. Cada proveedor tiene un número de proveedor (S#), único para dicho proveedor; un nombre (SNAME), no necesariamente único (a pesar de que los nombres que aparecen en el ejemplo son únicos); una calificación o valor de estado (STATUS); y una ubicación (CITY). Se asume que cada proveedor está ubicado en exactamente una ciudad.

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

2. P (Partes)

La tabla P representa las partes (para ser exactos, clase de parte). Cada clase de parte tiene un número de parte (P#), que es único; un nombre de parte (PNAME); un color (COLOR); un peso (WEIGHT) y una ubicación donde las partes de ese tipo se venden (CITY). Se asume donde ocurra alguna diferencia que los pesos de las partes están en libras. También se asume que cada clase de parte viene exactamente con un color y se guarda en un almacén en exactamente una ciudad.

P#	PNAME	COLOR	WEIGHT	CITY
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London
P5	Cam	Blue	12	Paris
P6	Cog	Red	19	London

3. SP (Envíos)

La Tabla SP representa envíos. Sirve a modo de conexión entre las otras dos tablas juntas. Por ejemplo, la primera fila de la tabla SP conecta a un proveedor específico de la tabla S (en este caso, proveedor S1) con una parte específica de la tabla P (en este caso, la parte P1) en otras palabras, representa un envío de partes de la clase P1 por el proveedor llamado S1 (y la cantidad de envío es 300). Así, cada envío tiene un número de proveedor (S#), un número de parte (P#) y una cantidad (QTY). Se asume que puede haber a lo más un envío en un momento dado para un proveedor y una parte dados; para un envío dado, asimismo, la combinación del valor S# y el valor de P# es única con respecto al conjunto de envíos que aparecen actualmente en la tabla SP.

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

4. J (Proyectos)

La Tabla J representa proyectos. Cuenta con un número de proyecto (J#), un nombre de proyecto (JNAME) y la ubicación donde se desarrolla el proyecto (CITY). Se considera como único el número de proyecto.

J#	JNAME	CITY
J1	Sorter	Paris
J2	Display	Rome
J3	OCR	Athens
J4	Console	Athens
J5	RAID	London
J6	EDS	Oslo
J7	Tape	London

5. SPJ (Envíos a Proyectos)

La Tabla SPJ representa envíos a proyectos. Cuenta con un número de proveedor (S#), un número de parte (P#), un número de proyecto (J#) y la cantidad enviada (QTY). Se considera como única la combinación de número de proveedor, número de parte y número de proyecto.

S#	P#	J#	QTY
S1	P1	J1	200
S1	P1	J4	700
S2	P3	J1	400
S2	P3	J2	200
S2	P3	J3	200
S2	P3	J4	500
S2	P3	J5	600
S2	P3	J6	400
S2	P3	J7	800
S2	P5	J2	100
S3	P3	J1	200
S3	P4	J2	500
S4	P6	J3	300
S4	P6	J7	300

S5	P2	J2	200
S5	P2	J4	100
S5	P5	J5	500
S5	P5	J7	100
S5	P6	J2	200
S5	P1	J4	100
S5	P3	J4	200
S5	P4	J4	800
S5	P5	J4	400
S5	P6	J4	500

IV. EJERCICIOS DE PROCEDIMIENTOS Y FUNCIONES

Considerar realizar los procedimientos y funciones necesarias para realizar las operaciones enunciadas con las tablas S, P, SP.

1. Obtenga el color y ciudad para las partes que no son de París, con un peso mayor de diez.

```

--- Ejercicio 01
SET serveroutput ON
CREATE OR REPLACE FUNCTION get_parts_color_city
RETURN VARCHAR2
IS
    v_result VARCHAR2(5000);
BEGIN
    v_result := '';

    FOR registro IN (
        SELECT color, city FROM P
        WHERE UPPER(city) <> 'PARIS'
        AND weight > 10
    ) LOOP
        v_result := v_result || 'Color: ' || registro.color
            || ' | Ciudad: ' || registro.city || CHR(10);
    END LOOP;

    IF v_result IS NULL THEN
        v_result := 'No existen resultados.';
    END IF;

    RETURN v_result;
EXCEPTION
    WHEN OTHERS THEN

```

```

        RETURN 'Error: ' || SQLERRM;
END get_parts_color_city;
/

```

```

BEGIN
    dbms_output.put_line(get_parts_color_city);
END;
/

```

```

Function GET_PARTS_COLOR_CITY compiled

No errors.
Color: Red| Ciudad: London
Color: Blue| Ciudad: Rome
Color: Red| Ciudad: London
Color: Red| Ciudad: London

PL/SQL procedure successfully completed.

```

2. Para todas las partes, obtenga el número de parte y el peso de dichas partes en gramos.

```

--- Ejercicio 02
SET serveroutput ON
CREATE OR REPLACE FUNCTION get_num_weight_parts
    RETURN VARCHAR2
IS
    v_result VARCHAR2(4000);
BEGIN
    v_result := '';

    FOR registro IN (
        SELECT p#, (weight*453.592) AS weight_gramos FROM P
    ) LOOP
        v_result := v_result || 'N° Parte: ' || registro.p# ||
            '| Peso (gr): ' || registro.weight_gramos || CHR(10);
    END LOOP;

    IF v_result IS NULL OR v_result = '' THEN
        v_result := 'No existen partes registradas.';
    END IF;

    RETURN v_result;
EXCEPTION
    WHEN OTHERS THEN
        RETURN 'Error: ' || SQLERRM;
END get_num_weight_parts;
/

```

```

BEGIN
    dbms_output.put_line(get_num_weight_parts);
END;
/

```

Function GET_NUM_WEIGHT_PARTS compiled

No errors.

Nº Parte: P1| Peso (gr): 5443,104

Nº Parte: P2| Peso (gr): 7711,064

Nº Parte: P3| Peso (gr): 7711,064

Nº Parte: P4| Peso (gr): 6350,288

Nº Parte: P5| Peso (gr): 5443,104

Nº Parte: P6| Peso (gr): 8618,248

PL/SQL procedure successfully completed.

3. Obtenga el detalle completo de todos los proveedores.

```

--- Ejercicio 03
SET serveroutput ON;
CREATE OR REPLACE PROCEDURE get_detalle_proveedor
IS
BEGIN
    FOR registro IN (
        SELECT s#, sname, status, city FROM S
    ) LOOP
        dbms_output.put_line('Proveedor: ' || registro.s# ||
            ' | Nombre: ' || registro.sname ||
            ' | Estado: ' || registro.status ||
            ' | Ciudad: ' || registro.city);
    END LOOP;
EXCEPTION
    WHEN OTHERS THEN
        dbms_output.put_line('Error: ' || SQLERRM);
END get_detalle_proveedor;
/

```

```

EXECUTE get_detalle_proveedor;

```



```

Procedure GET_DETALLES_PROVEEDOR compiled

No errors.
Proveedor: S1 | Nombre: Smith | Estado: 20 | Ciudad: London
Proveedor: S2 | Nombre: Jones | Estado: 10 | Ciudad: Paris
Proveedor: S3 | Nombre: Blake | Estado: 30 | Ciudad: Paris
Proveedor: S4 | Nombre: Clark | Estado: 20 | Ciudad: London
Proveedor: S5 | Nombre: Adams | Estado: 30 | Ciudad: Athens

PL/SQL procedure successfully completed.

```

4. Obtenga todas las combinaciones de proveedores y partes para aquellos proveedores y partes colocalizados.

```

--- Ejercicio 04
SET serveroutput ON;
CREATE OR REPLACE PROCEDURE get_combinacion_proovedores_partes
IS
BEGIN
    FOR registro IN (
        SELECT s.sname, p.pname , s.city FROM S s
        JOIN P p ON p.city = s.city
        ORDER BY s.city
    ) LOOP
        dbms_output.put_line('Ciudad: ' || registro.city ||
            ' | P: ' || registro.pname ||
            ' | S: ' || registro.sname);
    END LOOP;
EXCEPTION
    WHEN OTHERS THEN
        dbms_output.put_line('Error: ' || SQLERRM);
END get_combinacion_proovedores_partes;
/

```

```

EXECUTE get_combinacion_proovedores_partes;

```

```

Procedure GET_COMBINACION_PROVEEDORES_PARTES compiled

No errors.
Ciudad: London | P: Nut | S: Smith
Ciudad: London | P: Nut | S: Clark
Ciudad: London | P: Screw | S: Smith
Ciudad: London | P: Cog | S: Smith
Ciudad: London | P: Screw | S: Clark
Ciudad: London | P: Cog | S: Clark
Ciudad: Paris | P: Cam | S: Blake
Ciudad: Paris | P: Cam | S: Jones
Ciudad: Paris | P: Bolt | S: Blake
Ciudad: Paris | P: Bolt | S: Jones

PL/SQL procedure successfully completed.

```

- Obtenga todos los pares de nombres de ciudades de tal forma que el proveedor localizado en la primera ciudad del par abastece una parte almacenada en la segunda ciudad del par.

```

--- Ejercicio 05
SET serveroutput ON;
CREATE OR REPLACE PROCEDURE get_provedores_ciudad
IS
BEGIN
    FOR registro IN (
        SELECT DISTINCT s.city AS ciudad_proveedor , p.city AS
ciudad_parte FROM S s
        JOIN SP sp ON s.s# = sp.s#
        JOIN P p ON sp.p# = p.p#
        ORDER BY s.city, p.city
    ) LOOP
        dbms_output.put_line('Proveedor en: ' ||
registro.ciudad_proveedor ||
        ' -> Abastece partes en: ' || registro.ciudad_parte);
    END LOOP;
EXCEPTION
    WHEN OTHERS THEN
        dbms_output.put_line('Error: ' || SQLERRM);
END get_provedores_ciudad;
/

```

```
EXECUTE get_provedores_ciudad;
```

```

Procedure GET_PROVEEDORES_CIUADAD compiled

No errors.
Proveedor en: London -> Abastece partes en: London
Proveedor en: London -> Abastece partes en: Paris
Proveedor en: London -> Abastece partes en: Rome
Proveedor en: Paris -> Abastece partes en: London
Proveedor en: Paris -> Abastece partes en: Paris

PL/SQL procedure successfully completed.

```

6. Obtenga todos los pares de número de proveedor tales que los dos proveedores del par estén colocados.

```

--- Ejercicio 06
SET serveroutput ON
CREATE OR REPLACE PROCEDURE get_proveedores
IS
BEGIN
    FOR registro IN(
        SELECT s1.s# AS proveedor1, s2.s# AS proveedor2, s1.city FROM
S s1
        JOIN S s2 ON s1.city = s2.city
        AND s1.s# < s2.S#
        ORDER BY s1.city
    ) LOOP
        dbms_output.put_line('Ciudad: ' || registro.city ||
        ' | Pares: (' || registro.proveedor1 ||
        ', ' || registro.proveedor2 || ')');
    END LOOP;
EXCEPTION
    WHEN OTHERS THEN
        dbms_output.put_line('Error: ' || SQLERRM);
END get_proveedores;
/

```

```
EXECUTE get_proveedores;
```

```

Procedure GET_PROVEEDORES compiled

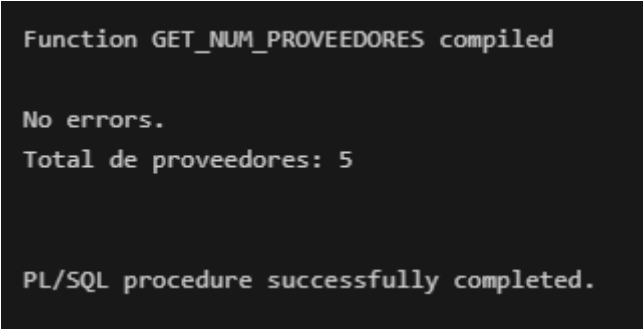
No errors.
Ciudad: London | Pares: (S1, S4)
Ciudad: Paris | Pares: (S2, S3)

```

7. Obtenga el número total de proveedores.

```
--- Ejercicio 07: Obtenga el número total de proveedores.
SET serveroutput ON;
CREATE OR REPLACE FUNCTION get_num_proveedores
    RETURN NUMBER
IS
    v_numero NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_numero FROM S;
    RETURN v_numero;
EXCEPTION
    WHEN OTHERS THEN
        RETURN -1;
END get_num_proveedores;
/
```

```
BEGIN
    dbms_output.put_line('Total de proveedores: ' ||
get_num_proveedores);
END;
/
```



```
Function GET_NUM_PROVEEDORES compiled

No errors.
Total de proveedores: 5

PL/SQL procedure successfully completed.
```

8. Obtenga la cantidad mínima y la cantidad máxima para la parte P2.

```
--- Ejercicio 08
SET serveroutput ON;
CREATE OR REPLACE PROCEDURE get_min_max_part
IS
    v_min NUMBER;
    v_max NUMBER;
BEGIN
    SELECT MIN(qty), MAX(qty)
        INTO v_min, v_max FROM SP
        WHERE UPPER(p#) = 'P2';

    dbms_output.put_line('P2 -> Cantidad mínima: ' ||
        v_min || ' - Cantidad máxima: ' || v_max);
```

```

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        dbms_output.put_line('No hay registro para la parte P2.');
```

```

    WHEN OTHERS THEN
        dbms_output.put_line('Error: ' || SQLERRM);
END get_min_max_part;
/
```

```
EXECUTE get_min_max_part;
```

```

Procedure GET_MIN_MAX_PART compiled

No errors.
P2 -> Cantidad mínima: 200 - Cantidad máxima: 400

PL/SQL procedure successfully completed.
```

9. Para cada parte abastecida, obtenga el número de parte y el total despachado.

```

--- Ejercicio 09
SET serveroutput ON;
CREATE OR REPLACE PROCEDURE get_parte_despachado
IS
BEGIN
    FOR registro IN (
        SELECT sp.p#, SUM(sp.qty) AS total FROM SP sp
        GROUP BY p#
        ORDER BY p#
    ) LOOP
        dbms_output.put_line('Parte: ' || registro.p# ||
            ' | Total: ' || registro.total);
    END LOOP;
EXCEPTION
    WHEN OTHERS THEN
        dbms_output.put_line('Error: ' || SQLERRM);
END get_parte_despachado;
/
```

```
EXECUTE get_parte_despachado;
```

```

Procedure GET_PARTE_DESPACHADO compiled

No errors.
Parte: P1 | Total: 600
Parte: P2 | Total: 1000
Parte: P3 | Total: 400
Parte: P4 | Total: 500
Parte: P5 | Total: 500
Parte: P6 | Total: 100

PL/SQL procedure successfully completed.

```

10. Obtenga el número de parte para todas las partes abastecidas por más de un proveedor.

```

--- Ejercicio 10
SET serveroutput ON
CREATE OR REPLACE PROCEDURE get_parte_proveedor
IS
BEGIN
    FOR registro IN (
        SELECT sp.p# , COUNT(DISTINCT s#) AS num_proveedores FROM SP
    sp
        GROUP BY p#
        HAVING COUNT(DISTINCT s#) > 1
        ORDER BY p#
    ) LOOP
        dbms_output.put_line('Parte: ' || registro.p# ||
            ' | N° Proveedores: ' || registro.num_proveedores);
    END LOOP;
EXCEPTION
    WHEN OTHERS THEN
        dbms_output.put_line('Error: ' || SQLERRM);
END get_parte_proveedor;
/

EXECUTE get_parte_proveedor;

```

```

Procedure GET_PARTE_PROVEEDOR compiled

No errors.

Parte: P1 | N° Proveedores: 2
Parte: P2 | N° Proveedores: 4
Parte: P4 | N° Proveedores: 2
Parte: P5 | N° Proveedores: 2

PL/SQL procedure successfully completed.

```

11. Obtenga el nombre de proveedor para todos los proveedores que abastecen la parte P2.

```

--- Ejercicio 11: Obtenga el nombre de proveedor para todos
--- los proveedores que abastecen la parte P2.
SET serveroutput ON;
CREATE OR REPLACE PROCEDURE get_proveedor_p2
IS
BEGIN
    FOR registro IN (
        SELECT DISTINCT s.sname FROM S s
        JOIN SP sp ON sp.s# = s.s#
        WHERE sp.p# = 'P2'
        ORDER BY s.sname
    ) LOOP
        dbms_output.put_line('Proveedor que abastece P2: ' ||
            registro.sname);
    END LOOP;
EXCEPTION
    WHEN OTHERS THEN
        dbms_output.put_line('Error: ' || SQLERRM);
END get_proveedor_p2;
/

```

```
EXECUTE get_proveedor_p2;
```

```

Procedure GET_PROVEEDOR_P2 compiled

No errors.
Proveedor que abastece P2: Blake
Proveedor que abastece P2: Clark
Proveedor que abastece P2: Jones
Proveedor que abastece P2: Smith

PL/SQL procedure successfully completed.

```

12. Obtenga el nombre de proveedor de quienes abastecen por lo menos una parte.

```

--- Ejercicio 12
SET serveroutput ON;
CREATE OR REPLACE PROCEDURE get_proveedor_un_parte
IS
BEGIN
    FOR registro IN (
        SELECT DISTINCT s.sname FROM SP sp
        JOIN S s ON sp.s# = s.s#
    ) LOOP
        dbms_output.put_line('Proveedor: ' ||
            registro.sname);
    END LOOP;
EXCEPTION
    WHEN OTHERS THEN
        dbms_output.put_line('Error: ' || SQLERRM);
END get_proveedor_un_parte;
/

```

```
EXECUTE get_proveedor_un_parte;
```

```

PL/SQL procedure successfully completed.

Proveedor: Smith
Proveedor: Jones
Proveedor: Blake
Proveedor: Clark

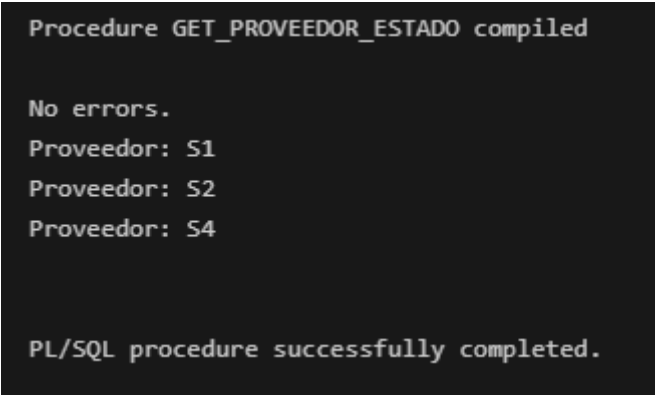
PL/SQL procedure successfully completed.

```


13. Obtenga el número de proveedor para los proveedores con estado menor que el máximo valor de estado en la tabla S.

```
--- Ejercicio 13
SET serveroutput ON;
CREATE OR REPLACE PROCEDURE get_proveedor_estado
IS
    v_max NUMBER;
BEGIN
    SELECT MAX(status) INTO v_max FROM S;
    FOR registro IN (
        SELECT s# FROM S
        WHERE status < v_max
    ) LOOP
        dbms_output.put_line('Proveedor: ' || registro.s#);
    END LOOP;
EXCEPTION
    WHEN OTHERS THEN
        dbms_output.put_line('Error: ' || SQLERRM);
END get_proveedor_estado;
/
```

```
EXECUTE get_proveedor_estado;
```



```
Procedure GET_PROVEEDOR_ESTADO compiled

No errors.
Proveedor: S1
Proveedor: S2
Proveedor: S4

PL/SQL procedure successfully completed.
```

14. Obtenga el nombre de proveedor para los proveedores que abastecen la parte P2 (aplicar EXISTS en su solución).

```
--- Ejercicio 14
SET serveroutput ON;
CREATE OR REPLACE PROCEDURE get_proveedor_abastece_p2
IS
BEGIN
    FOR registro IN (
        SELECT s.sname FROM S s
        WHERE EXISTS (SELECT 1 FROM SP sp
            WHERE sp.s# = s.s# AND UPPER(sp.p#) = 'P2')
    ) LOOP
        dbms_output.put_line('Proveedor: ' || registro.sname);
    END LOOP;
END get_proveedor_abastece_p2;
```

```

    ) LOOP
        dbms_output.put_line('Proveedor que abastece P2: ' ||
registro.sname);
    END LOOP;
EXCEPTION
    WHEN OTHERS THEN
        dbms_output.put_line('Error: ' || SQLERRM);
END get_proveedor_abastece_p2;
/

```

```
EXECUTE get_proveedor_abastece_p2;
```

```
Procedure GET_PROVEEDOR_ABASTECE_P2 compiled
```

```
No errors.
```

```
Proveedor que abastece P2: Smith
```

```
Proveedor que abastece P2: Jones
```

```
Proveedor que abastece P2: Blake
```

```
Proveedor que abastece P2: Clark
```

```
PL/SQL procedure successfully completed.
```

15. Obtenga el nombre de proveedor para los proveedores que no abastecen la parte P2.

```

SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE get_suppliers_not_p2
IS
BEGIN
    FOR r IN (
        SELECT s.sname
        FROM S s
        WHERE NOT EXISTS (
            SELECT 1
            FROM SP sp
            WHERE sp.s# = s.s#
            AND UPPER(sp.p#) = 'P2'
        )
    )
    LOOP
        DBMS_OUTPUT.PUT_LINE('Proveedor que NO abastece P2: ' ||
r.sname);
    END LOOP;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END get_suppliers_not_p2;
/

```

```
EXECUTE get_suppliers_not_p2;
```

```
Procedure GET_SUPPLIERS_NOT_P2 compiled

No errors.
Proveedor que NO abastece P2: Adams

PL/SQL procedure successfully completed.
```

16. Obtenga el nombre de proveedor para los proveedores que abastecen todas las partes.

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE get_suppliers_all_parts
IS
BEGIN
    FOR r IN (
        SELECT s.sname
        FROM S s
        WHERE NOT EXISTS (
            SELECT 1
            FROM P p
            WHERE NOT EXISTS (
                SELECT 1
                FROM SP sp
                WHERE sp.s# = s.s#
                AND sp.p# = p.p#
            )
        )
    )
    LOOP
        DBMS_OUTPUT.PUT_LINE('Proveedor que abastece TODAS las
partes: ' || r.sname);
    END LOOP;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END get_suppliers_all_parts;
/
```

```
EXECUTE get_suppliers_all_parts;
```

```
Procedure GET_SUPPLIERS_ALL_PARTS compiled

No errors.
Proveedor que abastece TODAS las partes: Smith

PL/SQL procedure successfully completed.
```

17. Obtenga el número de parte para todas las partes que pesan más de 16 libras o son abastecidas por el proveedor S2, ó cumplen con ambos criterios.

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE get_parts_weight_or_s2
IS
BEGIN
    FOR r IN (
        SELECT DISTINCT p.p#
        FROM P p
        LEFT JOIN SP sp ON p.p# = sp.p#
        WHERE p.weight > 16
            OR UPPER(sp.s#) = 'S2'
    )
    LOOP
        DBMS_OUTPUT.PUT_LINE('Parte #: ' || r.p#);
    END LOOP;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END get_parts_weight_or_s2;
/
```

```
EXECUTE get_parts_weight_or_s2;
```

```
Procedure GET_PARTS_WEIGHT_OR_S2 compiled

No errors.
Parte #: P1
Parte #: P2
Parte #: P3
Parte #: P6

PL/SQL procedure successfully completed.
```