

TAREA DE LABORATORIO

Alumno: Rodrigo Davalos
Código: 22200233

Creación de Tablas

Crear tabla de proveedores (S)

```
CREATE TABLE S (
    S# VARCHAR2(5) PRIMARY KEY,
    SNAME VARCHAR2(30),
    STATUS NUMBER,
    CITY VARCHAR2(20)
);
```

```
INSERT INTO S VALUES ('S1', 'Smith', 20, 'London');
INSERT INTO S VALUES ('S2', 'Jones', 10, 'Paris');
INSERT INTO S VALUES ('S3', 'Blake', 30, 'Paris');
INSERT INTO S VALUES ('S4', 'Clark', 20, 'London');
INSERT INTO S VALUES ('S5', 'Adams', 30, 'Athens');
COMMIT;
```

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

Crear tabla de partes (P)

```
CREATE TABLE P (
    P# VARCHAR2(5) PRIMARY KEY,
    PNAME VARCHAR2(30),
    COLOR VARCHAR2(15),
    WEIGHT NUMBER,
    CITY VARCHAR2(20)
);
```

```
INSERT INTO P VALUES ('P1', 'Nut', 'Red', 12, 'London');
```

```

INSERT INTO P VALUES ('P2', 'Bolt', 'Green', 17, 'Paris');
INSERT INTO P VALUES ('P3', 'Screw', 'Blue', 17, 'Rome');
INSERT INTO P VALUES ('P4', 'Screw', 'Red', 14, 'London');
INSERT INTO P VALUES ('P5', 'Cam', 'Blue', 12, 'Paris');
INSERT INTO P VALUES ('P6', 'Cog', 'Red', 19, 'London');
COMMIT;

```

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

Crear tabla de envíos (SP)

```

CREATE TABLE SP (
    S# VARCHAR2(5),
    P# VARCHAR2(5),
    QTY NUMBER,
    CONSTRAINT pk_sp PRIMARY KEY (S#, P#),
    CONSTRAINT fk_sp_s FOREIGN KEY (S#) REFERENCES S(S#),
    CONSTRAINT fk_sp_p FOREIGN KEY (P#) REFERENCES P(P#)
);

```

```

INSERT INTO SP VALUES ('S1', 'P1', 300);
INSERT INTO SP VALUES ('S1', 'P2', 200);
INSERT INTO SP VALUES ('S1', 'P3', 400);
INSERT INTO SP VALUES ('S1', 'P4', 200);
INSERT INTO SP VALUES ('S1', 'P5', 100);
INSERT INTO SP VALUES ('S1', 'P6', 100);
INSERT INTO SP VALUES ('S2', 'P1', 300);
INSERT INTO SP VALUES ('S2', 'P2', 400);
INSERT INTO SP VALUES ('S3', 'P2', 200);
INSERT INTO SP VALUES ('S4', 'P2', 200);
INSERT INTO SP VALUES ('S4', 'P4', 300);
INSERT INTO SP VALUES ('S4', 'P5', 400);
COMMIT;

```

```
SELECT * FROM SP
;
```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 12 en 0

	S#	P#	QTY
1	S1	P1	300
2	S1	P2	200
3	S1	P3	400
4	S1	P4	200
5	S1	P5	100
6	S1	P6	100
7	S2	P1	300
8	S2	P2	400
9	S3	P2	200
10	S4	P2	200
11	S4	P4	300
12	S4	P5	400

Crear tabla de proyectos (J)

```
CREATE TABLE J (
    J# VARCHAR2(5) PRIMARY KEY,
    JNAME VARCHAR2(30),
    CITY VARCHAR2(20)
);
```

```
INSERT INTO J VALUES ('J1', 'Sorter', 'Paris');
INSERT INTO J VALUES ('J2', 'Display', 'Rome');
INSERT INTO J VALUES ('J3', 'OCR', 'Athens');
INSERT INTO J VALUES ('J4', 'Console', 'Athens');
INSERT INTO J VALUES ('J5', 'RAID', 'London');
INSERT INTO J VALUES ('J6', 'EDS', 'Oslo');
INSERT INTO J VALUES ('J7', 'Tape', 'London');
COMMIT;
```

Hoja de Trabajo Generador de Consultas

```
SELECT * FROM J
;
```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 7 en 0.00

J#	JNAME	CITY
1 J1	Sorter	Paris
2 J2	Display	Rome
3 J3	OCR	Athens
4 J4	Console	Athens
5 J5	RAID	London
6 J6	EDS	Oslo
7 J7	Tape	London

Crear tabla de envíos a proyectos (SPJ)

```
CREATE TABLE SPJ (
    S# VARCHAR2(5),
    P# VARCHAR2(5),
    J# VARCHAR2(5),
    QTY NUMBER,
    CONSTRAINT pk_spj PRIMARY KEY (S#, P#, J#),
    CONSTRAINT fk_spj_s FOREIGN KEY (S#) REFERENCES S(S#),
    CONSTRAINT fk_spj_p FOREIGN KEY (P#) REFERENCES P(P#),
    CONSTRAINT fk_spj_j FOREIGN KEY (J#) REFERENCES J(J#)
);
```

```
INSERT INTO SPJ VALUES ('S1', 'P1', 'J1', 200);
INSERT INTO SPJ VALUES ('S1', 'P1', 'J4', 700);
INSERT INTO SPJ VALUES ('S2', 'P3', 'J1', 400);
INSERT INTO SPJ VALUES ('S2', 'P3', 'J2', 200);
INSERT INTO SPJ VALUES ('S2', 'P3', 'J3', 200);
INSERT INTO SPJ VALUES ('S2', 'P3', 'J4', 500);
INSERT INTO SPJ VALUES ('S2', 'P3', 'J5', 600);
INSERT INTO SPJ VALUES ('S2', 'P3', 'J6', 400);
INSERT INTO SPJ VALUES ('S2', 'P3', 'J7', 800);
INSERT INTO SPJ VALUES ('S2', 'P5', 'J2', 100);
INSERT INTO SPJ VALUES ('S3', 'P3', 'J1', 200);
INSERT INTO SPJ VALUES ('S3', 'P4', 'J2', 500);
INSERT INTO SPJ VALUES ('S4', 'P6', 'J3', 300);
INSERT INTO SPJ VALUES ('S4', 'P6', 'J7', 300);
INSERT INTO SPJ VALUES ('S5', 'P2', 'J2', 200);
INSERT INTO SPJ VALUES ('S5', 'P2', 'J4', 100);
```

```
INSERT INTO SPJ VALUES ('S5', 'P5', 'J5', 500);
INSERT INTO SPJ VALUES ('S5', 'P5', 'J7', 100);
COMMIT
```

Hoja de Trabajo Generador de Consultas

```
SELECT * FROM SPJ;
```

Salida de Script Resultado de la Consulta

SQL | Todas las Filas Recuperadas: 18 en 0.003 segundos

	S#	P#	J#	QTY
1	S1	P1	J1	200
2	S1	P1	J4	700
3	S2	P3	J1	400
4	S2	P3	J2	200
5	S2	P3	J3	200
6	S2	P3	J4	500
7	S2	P3	J5	600
8	S2	P3	J6	400
9	S2	P3	J7	800
10	S2	P5	J2	100
11	S3	P3	J1	200
12	S3	P4	J2	500
13	S4	P6	J3	300
14	S4	P6	J7	300
15	S5	P2	J2	200
16	S5	P2	J4	100
17	S5	P5	J5	500
18	S5	P5	J7	100

1. Ejercicios:

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

SET SERVEROUTPUT ON;

```
CREATE OR REPLACE PROCEDURE partes_no_paris AS
BEGIN
  FOR r IN (
    SELECT COLOR, CITY
    FROM P
    WHERE CITY <> 'Paris'
      AND WEIGHT > 10
  ) LOOP
    DBMS_OUTPUT.PUT_LINE('Color: ' || r.COLOR || ' - Ciudad: ' || r.CITY);
  END LOOP;
END;
/
```

```
EXEC partes_no_paris;
```

```
Procedure PARTES_NO_PARIS compilado

Color: Red - Ciudad: London
Color: Blue - Ciudad: Rome
Color: Red - Ciudad: London
Color: Red - Ciudad: London

Procedimiento PL/SQL terminado correctamente.
```

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

```
CREATE OR REPLACE FUNCTION peso_gramos(libras NUMBER)
RETURN NUMBER IS
BEGIN
    RETURN libras * 453.592; -- Conversión de libras a gramos
END;
/
```

```
CREATE OR REPLACE PROCEDURE partes_en_gramos AS
BEGIN
    FOR r IN (SELECT P#, WEIGHT FROM P) LOOP
        DBMS_OUTPUT.PUT_LINE('Parte: ' || r.P# ||
            ' -> ' || peso_gramos(r.WEIGHT) || ' gramos');
    END LOOP;
END;
/
```

```
EXEC partes_en_gramos;
```

```
Procedure PARTES_EN_GRAMOS compilado

Parte: P1 -> 5443,104 gramos
Parte: P2 -> 7711,064 gramos
Parte: P3 -> 7711,064 gramos
Parte: P4 -> 6350,288 gramos
Parte: P5 -> 5443,104 gramos
Parte: P6 -> 8618,248 gramos

Procedimiento PL/SQL terminado correctamente.
```

1.3. Obtenga el detalle completo de todos los proveedores.

```
CREATE OR REPLACE PROCEDURE mostrar_proveedores IS
BEGIN
FOR r IN (SELECT S#, SNAME, STATUS, CITY FROM S) LOOP
DBMS_OUTPUT.PUT_LINE('S#: ' || r.S# ||
'| Nombre: ' || r.SNAME ||
'| Status: ' || r.STATUS ||
'| Ciudad: ' || r.CITY);
END LOOP;
END;
/
EXEC mostrar_proveedores;
```

```
Procedure MOSTRAR_PROVEEDORES compilado

S#: S1 | Nombre: Smith | Status: 20 | Ciudad: London
S#: S2 | Nombre: Jones | Status: 10 | Ciudad: Paris
S#: S3 | Nombre: Blake | Status: 30 | Ciudad: Paris
S#: S4 | Nombre: Clark | Status: 20 | Ciudad: London
S#: S5 | Nombre: Adams | Status: 30 | Ciudad: Athens

Procedimiento PL/SQL terminado correctamente.
```

1.4. Obtenga todas las combinaciones de proveedores y partes para aquellos proveedores y partes co-localizados.

```
CREATE OR REPLACE PROCEDURE mostrar_partes IS
```

```

BEGIN
  FOR r IN (SELECT P#, PNAME, COLOR, WEIGHT, CITY FROM P) LOOP
    DBMS_OUTPUT.PUT_LINE('P#: ' || r.P# ||
      '| Nombre: ' || r.PNAME ||
      '| Color: ' || r.COLOR ||
      '| Peso: ' || r.WEIGHT ||
      '| Ciudad: ' || r.CITY);
  END LOOP;
END;
/

```

```

EXEC mostrar_partes;
Procedure MOSTRAR_PARTES compilado

P#: P1 | Nombre: Nut | Color: Red | Peso: 12 | Ciudad: London
P#: P2 | Nombre: Bolt | Color: Green | Peso: 17 | Ciudad: Paris
P#: P3 | Nombre: Screw | Color: Blue | Peso: 17 | Ciudad: Rome
P#: P4 | Nombre: Screw | Color: Red | Peso: 14 | Ciudad: London
P#: P5 | Nombre: Cam | Color: Blue | Peso: 12 | Ciudad: Paris
P#: P6 | Nombre: Cog | Color: Red | Peso: 19 | Ciudad: London

Procedimiento PL/SQL terminado correctamente.

```

- 1.5. 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.**

```

CREATE OR REPLACE PROCEDURE pares_ciudades IS
BEGIN
  FOR r IN (
    SELECT DISTINCT s.city AS ciudad_proveedor, p.city AS ciudad_parte
    FROM s
    JOIN spj ON s.s# = spj.s#
    JOIN p ON p.p# = spj.p#
  ) LOOP
    DBMS_OUTPUT.PUT_LINE('Proveedor en: ' || r.ciudad_proveedor ||
      ' -> Parte en: ' || r.ciudad_parte);
  END LOOP;
END;
/

```

```
EXEC pares_ciudades;
```

```
Procedure PARES_CIUDADES compilado

Proveedor en: London -> Parte en: London
Proveedor en: Athens -> Parte en: Paris
Proveedor en: Paris -> Parte en: Rome
Proveedor en: Paris -> Parte en: London
Proveedor en: Paris -> Parte en: Paris

Procedimiento PL/SQL terminado correctamente.
```

1.6. Obtenga todos los pares de número de proveedor tales que los dos proveedores del par estén co-localizados.

```
CREATE OR REPLACE PROCEDURE proveedores_colocalizados IS
BEGIN
    FOR r 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#
    ) LOOP
        DBMS_OUTPUT.PUT_LINE('Proveedor1: ' || r.proveedor1 ||
            ' | Proveedor2: ' || r.proveedor2 ||
            ' | Ciudad: ' || r.city);
    END LOOP;
END;
/
```

```
EXEC proveedores_colocalizados;
```

```
Procedure PROVEEDORES_COLOCALIZADOS compilado

Proveedor1: S2 | Proveedor2: S3 | Ciudad: Paris
Proveedor1: S1 | Proveedor2: S4 | Ciudad: London

Procedimiento PL/SQL terminado correctamente.
```

1.7. Obtenga el número total de proveedores.

```
CREATE OR REPLACE PROCEDURE total_proveedores IS
    v_total NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_total FROM S;
    DBMS_OUTPUT.PUT_LINE('Número total de proveedores: ' || v_total);
END;
/
EXEC total_proveedores;
```

```
Procedure TOTAL_PROVEEDORES compilado
Número total de proveedores: 5
Procedimiento PL/SQL terminado correctamente.
```

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

```
CREATE OR REPLACE PROCEDURE min_max_p2 IS
    v_min_qty NUMBER;
    v_max_qty NUMBER;
BEGIN
    SELECT MIN(QTY), MAX(QTY)
    INTO v_min_qty, v_max_qty
    FROM SPJ
    WHERE P# = 'P2';

    DBMS_OUTPUT.PUT_LINE('Parte P2 -> Cantidad mínima: ' || v_min_qty ||
        ' | Cantidad máxima: ' || v_max_qty);
END;
/
EXEC min_max_p2;
```

```
Procedure MIN_MAX_P2 compilado

Parte P2 -> Cantidad mínima: 100 | Cantidad máxima: 200

Procedimiento PL/SQL terminado correctamente.
```

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

```
CREATE OR REPLACE PROCEDURE total_despachado_por_parte IS
```

```
BEGIN
```

```
FOR r IN (
```

```
    SELECT P#, SUM(QTY) AS total_despachado
```

```
    FROM SPJ
```

```
    GROUP BY P#
```

```
) LOOP
```

```
    DBMS_OUTPUT.PUT_LINE('Parte: ' || r.P# ||
        ' | Total despachado: ' || r.total_despachado);
```

```
END LOOP;
```

```
END;
```

```
/
```

```
EXEC total_despachado_por_parte;
```

```
Procedure TOTAL_DESPACHADO POR PARTE compilado

Parte: P1 | Total despachado: 900
Parte: P3 | Total despachado: 3300
Parte: P5 | Total despachado: 700
Parte: P4 | Total despachado: 500
Parte: P6 | Total despachado: 600
Parte: P2 | Total despachado: 300
```

```
Procedimiento PL/SQL terminado correctamente.
```

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

```
CREATE OR REPLACE PROCEDURE partes_mas_de_un_proveedor IS
```

```
BEGIN
```

```

FOR r IN (
    SELECT P#
    FROM SPJ
    GROUP BY P#
    HAVING COUNT(DISTINCT S#) > 1
) LOOP
    DBMS_OUTPUT.PUT_LINE('Parte abastecida por más de un proveedor: ' || r.P#);
END LOOP;
END;
/

```

EXEC partes_mas_de_un_proveedor;

```

Procedure PARTES_MAS_DE_UN_PROVEEDOR compilado

Parte abastecida por más de un proveedor: P3
Parte abastecida por más de un proveedor: P5

Procedimiento PL/SQL terminado correctamente.

```

- 1.11. Obtenga el nombre de proveedor para todos los proveedores que abastecen la parte P2.**

```

CREATE OR REPLACE PROCEDURE proveedores_de_p2 IS
BEGIN
    FOR r IN (
        SELECT DISTINCT S.SNAME
        FROM S
        JOIN SPJ ON S.S#= SPJ.S#
        WHERE SPJ.P# = 'P2'
    ) LOOP
        DBMS_OUTPUT.PUT_LINE('Proveedor que abastece la parte P2: ' || r.SNAME);
    END LOOP;
END;
/

```

EXEC proveedores_de_p2;

```
Procedure PROVEEDORES_DE_P2 compilado

Proveedor que abastece la parte P2: Adams

Procedimiento PL/SQL terminado correctamente.
```

- 1.12. Obtenga el nombre de proveedor de quienes abastecen por lo menos una parte.**

```
CREATE OR REPLACE PROCEDURE proveedores_con_alguna_parte IS
BEGIN
FOR r IN (
    SELECT DISTINCT S.SNAME
    FROM S
    WHERE S.S# IN (SELECT S# FROM SPJ)
) LOOP
    DBMS_OUTPUT.PUT_LINE('Proveedor con al menos una parte: ' || r.SNAME);
END LOOP;
END;
/
```

```
EXEC proveedores_con_alguna_parte;
Procedure PROVEEDORES_CON_ALGUNA_PARTE compilado

Proveedor con al menos una parte: Smith
Proveedor con al menos una parte: Jones
Proveedor con al menos una parte: Blake
Proveedor con al menos una parte: Clark
Proveedor con al menos una parte: Adams

Procedimiento PL/SQL terminado correctamente.
```

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

```
CREATE OR REPLACE PROCEDURE proveedores_estado_menor_max IS
```

```

BEGIN
  FOR r IN (
    SELECT S#
    FROM S
    WHERE STATUS < (SELECT MAX(STATUS) FROM S)
  ) LOOP
    DBMS_OUTPUT.PUT_LINE('Proveedor con estado menor al máximo: ' || r.S#);
  END LOOP;
END;
/

```

```

EXEC proveedores_estado_menor_max;
Procedure PROVEEDORES_ESTADO_MENOR_MAX compilado

Proveedor con estado menor al máximo: S1
Proveedor con estado menor al máximo: S2
Proveedor con estado menor al máximo: S4

Procedimiento PL/SQL terminado correctamente.

```

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

```

CREATE OR REPLACE PROCEDURE proveedores_de_p2_exists IS
BEGIN
  FOR r IN (
    SELECT SNAME
    FROM S
    WHERE EXISTS (
      SELECT 1
      FROM SPJ
      WHERE SPJ.S#=S.S#
      AND SPJ.P#='P2'
    )
  ) LOOP
    DBMS_OUTPUT.PUT_LINE('Proveedor que abastece la parte P2 (EXISTS): ' || r.SNAME);
  END LOOP;
END;
/

```

EXEC proveedores_de_p2_exists;

```
Procedure PROVEEDORES_DE_P2_EXISTS compilado

Proveedor que abastece la parte P2 (EXISTS): Adams

Procedimiento PL/SQL terminado correctamente.
```

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

```
CREATE OR REPLACE PROCEDURE proveedores_no_abastecen_p2 IS
BEGIN
    FOR r IN (
        SELECT SNAME
        FROM S
        WHERE S# NOT IN (
            SELECT S#
            FROM SPJ
            WHERE P#= 'P2'
        )
    ) LOOP
        DBMS_OUTPUT.PUT_LINE('Proveedor que NO abastece la parte P2: ' ||
r.SNAME);
    END LOOP;
END;
/
```

```
EXEC proveedores_no_abastecen_p2;
```

```
Procedure PROVEEDORES_NO_ABASTECEN_P2 compilado

Proveedor que NO abastece la parte P2: Jones
Proveedor que NO abastece la parte P2: Clark
Proveedor que NO abastece la parte P2: Smith
Proveedor que NO abastece la parte P2: Blake

Procedimiento PL/SQL terminado correctamente.
```

- 1.16. Obtenga el nombre de proveedor para los proveedores que abastecen todas las partes.**

```
CREATE OR REPLACE PROCEDURE proveedores_abastecen_todas_partes IS
BEGIN
    FOR r IN (
        SELECT SNAME
        FROM S
        WHERE NOT EXISTS (
            SELECT P#
            FROM P
            WHERE P# NOT IN (
                SELECT SPJ.P#
                FROM SPJ
                WHERE SPJ.S# = S.S#
            )
        )
    ) LOOP
        DBMS_OUTPUT.PUT_LINE('Proveedor que abastece todas las partes: ' || r.SNAME);
    END LOOP;
END;
/
```

```
EXEC proveedores_abastecen_todas_partes;
```

```
Procedure PROVEEDORES_ABASTECEN_TODAS_PARTES compilado
Procedimiento PL/SQL terminado correctamente.
```

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

```
CREATE OR REPLACE PROCEDURE partes_peso_o_proveedor IS
BEGIN
    FOR r IN (
        SELECT DISTINCT P#
        FROM P
        WHERE WEIGHT > 16
        OR P# IN (
            SELECT P#
            FROM SPJ
        )
    ) LOOP
        DBMS_OUTPUT.PUT_LINE('Número de parte: ' || r.P#);
    END LOOP;
END;
/
```

```
        WHERE S# = 'S2'  
    )  
) LOOP  
    DBMS_OUTPUT.PUT_LINE('Parte que cumple condición: ' || r.P#);  
END LOOP;  
END;  
/  
  
EXEC partes_peso_o_proveedor;
```

```
Procedure PARTES_PESO_O_PROVEEDOR compilado
```

```
Parte que cumple condición: P2  
Parte que cumple condición: P3  
Parte que cumple condición: P5  
Parte que cumple condición: P6
```

```
Procedimiento PL/SQL terminado correctamente.
```