- **Actividad 1.**

Queremos crear los siguientes subprogramas (teniendo en cuenta las excepciones que se puedan producir en la ejecución de cada uno y tratándolas o propagándolas como corresponda).

1. Función que compruebe en la tabla pedidos si existe un pedido con el número que se le pase. Devolverá verdadero o falso.

```
CREATE OR REPLACE FUNCTION existe_pedido(p_num NUMBER) RETURN BOOLEAN IS
v_count NUMBER;
BEGIN
SELECT COUNT(*)
INTO v_count
FROM PEDIDOS
WHERE NUM = p_num;

RETURN v_count > 0;
EXCEPTION
WHEN NO_DATA_FOUND THEN
RETURN FALSE;
WHEN OTHERS THEN
RAISE;
END;
```

Y se comprueba con un bloque PL/SQL

```
BEGIN
IF existe_pedido(1) THEN
DBMS_OUTPUT.PUT_LINE('El pedido 1 existe.');
ELSE
DBMS_OUTPUT.PUT_LINE('El pedido 1 no existe.');
END IF;
END;
```

2. Función que devuelve los datos de un pedido (de la tabla pedidos) cuyo número se le pasa.

```
CREATE OR REPLACE FUNCTION obtener_pedido(p_num_pedido NUMBER)
RETURN PEDIDOS%ROWTYPE
IS
v_pedido PEDIDOS%ROWTYPE;
BEGIN
SELECT * INTO v_pedido
FROM PEDIDOS
WHERE NUM = p_num_pedido;
RETURN v_pedido;
EXCEPTION
WHEN NO_DATA_FOUND THEN
RAISE_APPLICATION_ERROR(-20001, 'No existe el pedido ' || p_num_pedido);
WHEN OTHERS THEN
RAISE_APPLICATION_ERROR(-20002, 'Error al obtener el pedido: ' || SQLERRM);
END obtener_pedido;
/
```

3. Procedimiento que muestra un listado con todas las líneas del pedido cuyo número se le pasa. En caso de no tener líneas deberá mostrar un mensaje indicándolo. El listado debe ser de la siguiente manera:

Nº Línea	CodigoProd	NombreProducto	Precio	Cantidad	Importe
:::::::::	:::::::::::	:::::::::::	:::::	:::::::	::::::

```
CREATE OR REPLACE PROCEDURE mostrar_lineas_pedido(p_num_pedido NUMBER)
IS
CURSOR c_lineas IS
SELECT l.NUM, l.PRODUCTO, p.NOMBRE, p.PRECIO, l.CANTIDAD, l.IMPORTE
FROM LINEAS l
JOIN PRODUCTOS p ON l.PRODUCTO = p.CODIGO
WHERE l.NUMPEDIDO = p_num_pedido
ORDER BY l.NUM;
v_count NUMBER := 0;
BEGIN
IF NOT existe_pedido(p_num_pedido) THEN
RAISE_APPLICATION_ERROR(-20003, 'No existe el pedido ' || p_num_pedido);
END IF;
DBMS_OUTPUT.PUT_LINE('Nº Línea CodigoProd NombreProducto Precio Cantidad Importe');
DBMS_OUTPUT.PUT_LINE('-------- ---------- -------------- ------ -------- -------');
FOR r_linea IN c_lineas LOOP
v_count := v_count + 1;
```

```
DBMS_OUTPUT.PUT_LINE(LPAD(r_linea.NUM, 8) || ' ' ||
LPAD(r_linea.PRODUCTO, 10) || ' ' ||
RPAD(r_linea.NOMBRE, 16) || ' ' ||
LPAD(TO_CHAR(r_linea.PRECIO, '990.99'), 6) || ' ' ||
LPAD(TO_CHAR(r_linea.CANTIDAD), 8) || ' ' ||
LPAD(TO_CHAR(r_linea.IMPORTE, '990.99'), 7));
END LOOP;
IF v_count = 0 THEN
DBMS_OUTPUT.PUT_LINE('El pedido ' || p_num_pedido || ' no tiene líneas.');
END IF;
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Error al mostrar líneas de pedido: ' || SQLERRM);
END mostrar_lineas_pedido;
/
```

4. Procedimiento que devuelve los datos de un cliente (de la tabla clientes) a partir del código de cliente. Debe incluir un parámetro (true/false) para informar si existe o no el cliente.

```
CREATE OR REPLACE PROCEDURE obtener_cliente(
p_cod_cliente IN VARCHAR2,
p_nombre OUT VARCHAR2,
p_apellidos OUT VARCHAR2,
p_edad OUT NUMBER,
p_existe OUT BOOLEAN
) IS
BEGIN
SELECT NOMBRE, APELLIDOS, EDAD
INTO p_nombre, p_apellidos, p_edad
FROM CLIENTES
WHERE CODIGO = p_cod_cliente;
p_existe := TRUE;
EXCEPTION
WHEN NO_DATA_FOUND THEN
p_existe := FALSE;
WHEN OTHERS THEN
p_existe := FALSE;
RAISE_APPLICATION_ERROR(-20004, 'Error al obtener el cliente: ' || SQLERRM);
END obtener_cliente;
```

```
/
```

5. Procedimiento o bloque anónimo que a partir de un número de pedido, si existe, nos muestre todos los datos del pedido, los del cliente y el listado de todas las líneas que tiene, **utilizando los subprogramas anteriores**.

```
CREATE OR REPLACE PROCEDURE mostrar_info_pedido(p_num_pedido NUMBER)
IS
v_pedido PEDIDOS%ROWTYPE;
v_nombre_cliente VARCHAR2(30);
v_apellidos_cliente VARCHAR2(30);
v_edad_cliente NUMBER;
v_existe BOOLEAN;
BEGIN
-- Comprobar si existe el pedido
IF NOT existe_pedido(p_num_pedido) THEN
DBMS_OUTPUT.PUT_LINE('No existe el pedido ' || p_num_pedido);
RETURN;
END IF;
-- Obtener datos del pedido
v_pedido := obtener_pedido(p_num_pedido);
-- Obtener datos del cliente
obtener_cliente(v_pedido.CLIENTE, v_nombre_cliente, v_apellidos_cliente, v_edad_cliente, v_existe);
-- Mostrar información del pedido
DBMS_OUTPUT.PUT_LINE('===== INFORMACIÓN DEL PEDIDO ' || p_num_pedido || ' =====');
DBMS_OUTPUT.PUT_LINE('Fecha: ' || TO_CHAR(v_pedido.FECHA, 'DD/MM/YYYY'));
DBMS_OUTPUT.PUT_LINE('Fecha prevista: ' || TO_CHAR(v_pedido.FECHA_PREVISTA,
'DD/MM/YYYY'));
DBMS_OUTPUT.PUT_LINE('Gastos de envío: ' || NVL(TO_CHAR(v_pedido.GASTOS_ENVIO, '990.99'),
'N/A'));
DBMS_OUTPUT.PUT_LINE('Total: ' || TO_CHAR(v_pedido.TOTAL, '990.99'));
-- Mostrar información del cliente
DBMS_OUTPUT.PUT_LINE('');
DBMS_OUTPUT.PUT_LINE('===== INFORMACIÓN DEL CLIENTE =====');
IF v_existe THEN
DBMS_OUTPUT.PUT_LINE('Código: ' || v_pedido.CLIENTE);
DBMS_OUTPUT.PUT_LINE('Nombre: ' || v_nombre_cliente || ' ' || v_apellidos_cliente);
DBMS_OUTPUT.PUT_LINE('Edad: ' || v_edad_cliente);
ELSE
DBMS_OUTPUT.PUT_LINE('No se encontró el cliente con código ' || v_pedido.CLIENTE);
END IF;
-- Mostrar líneas del pedido
DBMS_OUTPUT.PUT_LINE('');
DBMS_OUTPUT.PUT_LINE('===== LÍNEAS DEL PEDIDO =====');
mostrar_lineas_pedido(p_num_pedido);
```

```
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Error al mostrar información del pedido: ' || SQLERRM);
END mostrar_info_pedido;
/
```

- **Actividad 2.**

Queremos controlar algunas restricciones mediante triggers: (debes crear un disparador para cada ejercicio):

1. Cada vez que se vaya a insertar o modificar una línea de un pedido debe actualizarse correctamente su importe (cantidad * precio del producto).

```
CREATE OR REPLACE TRIGGER trg_linea_importe
BEFORE INSERT OR UPDATE
ON LINEAS
FOR EACH ROW
DECLARE
v_precio NUMBER;
BEGIN
SELECT PRECIO
INTO v_precio
FROM PRODUCTOS
WHERE CODIGO = :NEW.PRODUCTO;

:NEW.IMPORTE := :NEW.CANTIDAD * v_precio;
END;
```

2. Cada vez que se inserten, se borren o modifiquen líneas hay que actualizar el importe total del pedido afectado.

```
CREATE OR REPLACE TRIGGER trg_actualiza_total
AFTER INSERT OR UPDATE OR DELETE
ON LINEAS
FOR EACH ROW
```

```
DECLARE
v_suma NUMBER;
v_pedido NUMBER;
BEGIN
v_pedido := NVL(:NEW.NUMPEDIDO, :OLD.NUMPEDIDO);

SELECT NVL(SUM(IMPORTE), 0)
INTO v_suma
FROM LINEAS
WHERE NUMPEDIDO = v_pedido;

UPDATE PEDIDOS
SET TOTAL = v_suma
WHERE NUM = v_pedido;
END;
```

3. Trigger para hacer que la clave ajena CLIENTE en la tabla pedidos se tenga modificación en cascada (modificación automática de la clave ajena si se modifica la clave primaria de un cliente).

```
CREATE OR REPLACE TRIGGER trg_update_cliente
AFTER UPDATE OF CODIGO
ON CLIENTES
FOR EACH ROW
BEGIN
UPDATE PEDIDOS
SET CLIENTE = :NEW.CODIGO
WHERE CLIENTE = :OLD.CODIGO;
END;
```