CONCEPTOS DE BASES DE DATOS II

Programador Universitario – Lic. en Informática – Ing. en Informática
Facultad de Ciencias Exactas y Tecnología – UNT

Trabajo Práctico Nro. 7 - Ciclo 2023

Functions

Ejercicio nro. 1:

a)

```
CREATE OR REPLACE FUNCTION empleado_modifica(
    p_dni character varying, p_campo character varying, p_fecha date)
RETURNS void AS $BODY$
DECLARE
    v_id_empleado integer;
    v_consulta character varying;
BEGIN
    IF NOT EXISTS(SELECT * FROM persona
                    INNER JOIN empleado ON id_persona = id_empleado
                    WHERE dni = p_dni) THEN
            RAISE EXCEPTION 'El empleado ingresado no existe.';
    END IF;
    v_id_empleado := (SELECT id_empleado FROM persona
                    INNER JOIN empleado ON id_persona = id_empleado
                    WHERE dni = p_dni);
    IF p_campo != 'fecha_ingreso' AND p_campo != 'fecha_baja' THEN
            RAISE EXCEPTION 'El campo ingresado solo puede ser fecha_ingreso/fecha_baja.';
    END IF;
    IF p_fecha IS NULL THEN
            RAISE EXCEPTION 'El campo fecha debe tener un valor válido';
    END IF;
    IF p_campo = 'fecha_baja' AND p_fecha < (SELECT fecha_ingreso
                                            FROM empleado
                                            WHERE id_empleado = v_id_empleado) THEN
            RAISE EXCEPTION 'La fecha de baja no puede ser anterior a la fecha de ingreso.';
    END IF;

    v_consulta = 'UPDATE empleado SET ' || p_campo || ' = ''' || p_fecha || ''' WHERE
id_empleado = ' || v_id_empleado;
    EXECUTE v_consulta;
    RAISE NOTICE 'Empleado modificado exitosamente';
END;
$BODY$ LANGUAGE plpgsql;
```

b)

```
CREATE OR REPLACE FUNCTION medicamento_modifica(
    p_tabla char, p_nombre character varying, p_ajuste char, p_porcentaje float)
RETURNS void AS $BODY$
BEGIN
```

```
    -- Validaciones
    IF p_tabla IS NULL OR (p_tabla != 'L' AND p_tabla != 'P' AND p_tabla != 'M') THEN
            RAISE EXCEPTION 'Debe ingresar la letra de la tabla en la cual buscar, valores permitidos: L/P/M';
    END IF;
    IF p_nombre IS NULL OR p_nombre = '' THEN
            RAISE EXCEPTION 'Debe ingresar un nombre';
    END IF;
    IF p_ajuste IS NULL OR (p_ajuste != 'A' AND p_ajuste != 'D') THEN
            RAISE EXCEPTION 'Debe ingresar el tipo de ajuste, valores permitidos: A, D';
    END IF;
    IF p_porcentaje < 0.01 OR p_porcentaje > 0.99 THEN
            RAISE EXCEPTION 'El porcentaje a ajustar debe estar entre [0.01, 0.99]';
    END IF;
    -- Validaciones y actualizaciones según tabla elegida
    IF p_tabla = 'L' THEN
            IF NOT EXISTS(SELECT * FROM laboratorios WHERE laboratorio = p_nombre) THEN
                    RAISE EXCEPTION 'El laboratorio ingresado no existe';
            END IF;
            UPDATE medicamento SET precio = precio * CASE
                            WHEN (p_ajuste = 'A') THEN (1 + p_porcentaje)
                            WHEN (p_ajuste = 'D') THEN (1 - p_porcentaje)
                            END
            WHERE id_laboratorio IN (SELECT id_laboratorio
                                    FROM laboratorios
                                    WHERE laboratorio = p_nombre);
            RAISE NOTICE 'Los precios de los medicamentos del laboratorio ingresado se modificaron correctamente';
    END IF;
    IF p_tabla = 'P' THEN
            IF NOT EXISTS(SELECT * FROM proveedores WHERE proveedor = p_nombre) THEN
                    RAISE EXCEPTION 'El proveedor ingresado no existe';
            END IF;
            UPDATE medicamento SET precio = precio * CASE
                            WHEN (p_ajuste = 'A') THEN (1 + p_porcentaje)
                            WHEN (p_ajuste = 'D') THEN (1 - p_porcentaje)
                            END
            WHERE id_medicamento IN (SELECT id_medicamento
                                    FROM compras
                                    INNER JOIN proveedores USING(id_proveedor)
                                    WHERE proveedor = p_nombre);
            RAISE NOTICE 'Los precios de los medicamentos del proveedor ingresado se modificaron correctamente';
    END IF;
```

```
    IF p_tabla = 'M' THEN
        IF NOT EXISTS(SELECT * FROM medicamento WHERE nombre = p_nombre) THEN
            RAISE EXCEPTION 'El medicamento ingresado no existe';
        END IF;
        UPDATE medicamento SET precio = precio * CASE
                WHEN (p_ajuste = 'A') THEN (1 + p_porcentaje)
                WHEN (p_ajuste = 'D') THEN (1 - p_porcentaje)
                END
        WHERE id_medicamento = (SELECT id_medicamento
                                FROM medicamento WHERE nombre = p_nombre);
        RAISE NOTICE 'Los precios del medicamento ingresado se modificaron
correctamente';
    END IF;
END;
$BODY$ LANGUAGE plpgsql;
```

c)

```
CREATE OR REPLACE FUNCTION cargo_abm(
    p_cargo character varying, p_accion character varying)
RETURNS void AS $BODY$
BEGIN
    IF p_cargo IS NULL OR p_cargo = '' THEN
        RAISE EXCEPTION 'Se debe indicar un nombre válido de cargo';
    END IF;
    IF p_accion IS NULL OR p_accion = '' THEN
        RAISE EXCEPTION 'Se debe indicar una acción';
    END IF;

    CASE p_accion
    WHEN 'insert' THEN
        IF EXISTS(SELECT * FROM cargo WHERE cargo = p_cargo) THEN
            RAISE EXCEPTION 'El cargo ingresado ya existe';
        ELSE
        INSERT INTO cargo VALUES ((SELECT MAX(id_cargo) + 1 FROM cargo), p_cargo);
            RAISE NOTICE 'El cargo se ingresó correctamente';
        END IF;
    WHEN 'delete' THEN
        DELETE FROM cargo WHERE cargo = p_cargo;
        IF NOT FOUND THEN
            RAISE EXCEPTION 'No se encontró el cargo para eliminarlo';
        ELSE
            RAISE NOTICE 'El cargo se eliminó correctamente';
        END IF;
```

```
        ELSE -- el Update
                IF NOT EXISTS(SELECT * FROM cargo WHERE cargo = p_cargo) THEN
                        RAISE EXCEPTION 'El cargo ingresado no existe';
                ELSE
                        UPDATE cargo SET cargo = p_accion WHERE cargo = p_cargo;
                        RAISE NOTICE 'El cargo se ha modificado correctamente';
                END IF;
        END CASE;
    END;
    $BODY$ LANGUAGE plpgsql;
```

d)

```
    CREATE OR REPLACE FUNCTION alta_tablas_sistema(
        p_tabla character varying, p_valor character varying)
    RETURNS void AS $BODY$
    DECLARE
        v_consulta character varying;
        v_anexo_tabla character varying;
    BEGIN
        IF p_tabla IS NULL OR (p_tabla != 'tipo_estudio' AND p_tabla != 'patologia'
                                AND p_tabla != 'clasificacion' AND p_tabla != 'especialidad') THEN
                RAISE EXCEPTION 'Se debe indicar un nombre de tabla válido, valores permitidos:
    tipo_estudio/patologia/clasificacion/especialidad';
        END IF;
        IF p_valor IS NULL OR p_valor = '' THEN
                RAISE EXCEPTION 'Se debe indicar un valor';
        END IF;
        -- validaciones según tabla
        v_anexo_tabla := '';
        CASE p_tabla
        WHEN 'tipo_estudio' THEN
                IF EXISTS(SELECT * FROM tipo_estudio WHERE tipo_estudio = p_valor) THEN
                        RAISE EXCEPTION 'El tipo de estudio ingresado ya existe';
                END IF;
                p_tabla := 'tipo';
                v_anexo_tabla := '_estudio';
        WHEN 'patologia' THEN
                IF EXISTS(SELECT * FROM patologia WHERE nombre = p_valor) THEN
                        RAISE EXCEPTION 'La patologia ingresada ya existe';
                END IF;
        WHEN 'clasificacion' THEN
                IF EXISTS(SELECT * FROM clasificacion WHERE clasificacion = p_valor) THEN
                        RAISE EXCEPTION 'La clasificación ingresada ya existe';
                END IF;
```

```
    WHEN 'especialidad' THEN
        IF EXISTS(SELECT * FROM especialidad WHERE especialidad = p_valor) THEN
            RAISE EXCEPTION 'La especialidad ingresada ya existe';
        END IF;
    END CASE;
    -- inserción
    v_consulta = 'INSERT INTO ' || p_tabla || v_anexo_tabla || ' VALUES((SELECT MAX(id_' ||
p_tabla || ') + 1 FROM ' || p_tabla || v_anexo_tabla || '), ''' || p_valor || ''')';
    EXECUTE v_consulta;
    RAISE NOTICE 'La información se ingresó correctamente';
END;
$BODY$ LANGUAGE plpgsql;
```

Ejercicio nro. 2:

Aquí se hace uso de los tipos definidos en el TP05 resuelto.

a)

```
CREATE OR REPLACE FUNCTION paciente_obra_social_listar(
        p_obra_social character varying)
RETURNS SETOF tipo_paciente_os AS $BODY$
BEGIN
        RETURN QUERY
                SELECT id_paciente, p.nombre, apellido, sigla, os.nombre
                FROM obra_social os
                INNER JOIN paciente USING(id_obra_social)
                INNER JOIN persona p ON id_persona = id_paciente
                WHERE os.nombre = p_obra_social;
END;
$BODY$ LANGUAGE plpgsql
```

b)

```
-- Tipo necesario
CREATE TYPE tipo_medicamento_proveedor AS (
        id INTEGER,
        nombre VARCHAR(50),
        clasificacion VARCHAR(75),
        laboratorio VARCHAR(50),
        proveedor VARCHAR(50),
        precio NUMERIC(10,2) );
```

```
CREATE OR REPLACE FUNCTION medicamento_listar_por_proveedor(
        p_proveedor character varying)
RETURNS SETOF tipo_medicamento_proveedor AS $BODY$
BEGIN
        RETURN QUERY
                SELECT id_medicamento, nombre, clasificacion, laboratorio, proveedor,
        precio_unitario
                FROM medicamento
                INNER JOIN laboratorio USING(id_laboratorio)
                INNER JOIN clasificacion USING(id_clasificacion)
                INNER JOIN compra USING(id_medicamento)
                INNER JOIN proveedor USING(id_proveedor)
                WHERE proveedor = p_proveedor;
END;
$BODY$ LANGUAGE plpgsql
```

c)

```
CREATE OR REPLACE FUNCTION consulta_listar_por_fecha(p_fecha date)
RETURNS SETOF tipo_consulta AS $BODY$
BEGIN
   RETURN QUERY
        SELECT pac.nombre, pac.apellido, emp.nombre, emp.apellido, fecha, c.nombre
                FROM persona pac
                INNER JOIN paciente ON id_persona = id_paciente
                INNER JOIN consulta USING (id_paciente)
                INNER JOIN persona emp ON emp.id_persona = id_empleado
                INNER JOIN consultorio c USING(id_consultorio)
                WHERE fecha = p_fecha;
END;
$BODY$ LANGUAGE plpgsql
```

d)

```
CREATE OR REPLACE FUNCTION internacion_por_paciente(p_dni character varying)
RETURNS SETOF tipo_internacion AS $BODY$
BEGIN
   RETURN QUERY
        SELECT pac.nombre, pac.apellido, emp.nombre, emp.apellido, costo, fecha_alta
                FROM persona pac
                INNER JOIN paciente ON id_persona = id_paciente
                INNER JOIN internacion USING(id_paciente)
                INNER JOIN persona emp ON emp.id_persona = ordena_internacion
                WHERE pac.dni = p_dni AND fecha_alta IS NOT NULL;
END;
$BODY$ LANGUAGE plpgsql
```

e)

```
CREATE OR REPLACE FUNCTION public.medicamento_por_laboratorio(
        p_laboratorio character varying)
  RETURNS SETOF tipo_medicamento AS $BODY$
BEGIN
        RETURN QUERY
                SELECT id_medicamento, nombre, stock, clasificacion, laboratorio
                FROM medicamento
                INNER JOIN laboratorio USING(id_laboratorio)
                INNER JOIN clasificacion USING(id_clasificacion)
                WHERE laboratorio = p_laboratorio;
END;
$BODY$ LANGUAGE 'plpgsql'
```

f)

```
CREATE OR REPLACE FUNCTION factura_listar_por_paciente(p_dni character varying)
RETURNS SETOF tipo_facturacion AS $BODY$
BEGIN
        RETURN QUERY
                SELECT id_factura, fecha, monto, nombre, apellido
                FROM persona
                INNER JOIN factura ON id_persona = id_paciente
                WHERE dni = p_dni;
END;
$BODY$ LANGUAGE plpgsql
```

g)

```
CREATE OR REPLACE FUNCTION mantenimiento_equipo_por_empleado
        (p_dni character varying)
RETURNS SETOF tipo_mantenimiento_equipo AS $BODY$
BEGIN
        RETURN QUERY
                SELECT p.nombre, apellido, e.nombre, marca, me.fecha_ingreso, estado
                FROM persona p
                INNER JOIN mantenimiento_equipo me ON id_persona = id_empleado
                INNER JOIN equipo e USING(id_equipo)
                WHERE dni = p_dni;
END;
$BODY$ LANGUAGE plpgsql
```

h)

```
CREATE TYPE tipo_facturacion_deuda AS (
        id BIGINT,
        fecha DATE,
        monto NUMERIC(10,2),
        nombre VARCHAR(100),
        apellido VARCHAR(100),
        deuda TEXT);

CREATE OR REPLACE FUNCTION factura_estado_deuda()
RETURNS SETOF tipo_facturacion_deuda AS $BODY$
BEGIN
        RETURN QUERY
                SELECT id_factura, fecha, monto, nombre, apellido,
                        CASE
                                WHEN saldo < 500000 THEN 'El cobro puede esperar'
                                WHEN saldo > 1000000 THEN 'Cobrar urgente'
                                ELSE 'Cobrar prioridad'
                        END AS deuda
                FROM factura
                INNER JOIN persona ON id_persona = id_paciente;
END;
$BODY$ LANGUAGE plpgsql
```

i)

```
CREATE TYPE tipo_tabla_sistema AS (
        id INTEGER,
        valor VARCHAR(100) );

CREATE OR REPLACE FUNCTION listar_tabla_sistema(p_tabla varchar)
        RETURNS SETOF tipo_tabla_sistema AS $BODY$
DECLARE
        reg tipo_tabla_sistema;
BEGIN
        IF p_tabla IS NULL OR (p_tabla != 'cargo' AND p_tabla != 'especialidad'
                AND p_tabla != 'clasificacion' AND p_tabla != 'patologia'
                AND p_tabla != 'tipoestudio') THEN
                RAISE EXCEPTION 'La tabla ingresada no es correcta';
        END IF;
        FOR reg IN EXECUTE 'SELECT * FROM ' || p_tabla LOOP
                RETURN NEXT reg;
        END LOOP;
END;
$BODY$ LANGUAGE plpgsql
```

CONCEPTOS DE BASES DE DATOS II

Programador Universitario – Lic. en Informática – Ing. en Informática
Facultad de Ciencias Exactas y Tecnología – UNT

Trabajo Práctico Nro. 7 - Ciclo 2023

Ejercicio nro. 3:

Plantee e implemente la o las funciones necesarias para realizar las siguientes tareas:

a)

```
CREATE OR REPLACE FUNCTION cama_arreglo(
        p_id_cama smallint, p_estado character varying,
        p_fecha date, p_dni character varying)
        RETURNS void AS $BODY$
DECLARE
        v_id_empleado integer;
BEGIN
        -- validaciones de parámetros
        IF p_id_cama IS NULL OR NOT EXISTS(SELECT * FROM mantenimiento_cama
                        WHERE id_cama = p_id_cama AND fecha_egreso IS NULL) THEN
                RAISE EXCEPTION 'La cama ingresada no está en mantenimiento';
        END IF;
        IF p_estado IS NULL OR p_estado = '' THEN
                RAISE EXCEPTION 'Debe ingresar un estado';
        END IF;
        IF p_fecha IS NULL THEN
                RAISE EXCEPTION 'Debe ingresar una fecha de finalización';
        END IF;
        IF p_dni IS NULL OR NOT EXISTS(SELECT * FROM persona
                                        INNER JOIN empleado ON id_persona = id_empleado
                                        WHERE dni = p_dni) THEN
                RAISE EXCEPTION 'El empleado ingresado no existe';
        END IF;
        IF p_fecha < (SELECT fecha_ingreso FROM mantenimiento_cama
                        WHERE id_cama = p_id_cama AND fecha_egreso IS NULL) THEN
           RAISE EXCEPTION 'La fecha de egreso no puede ser anterior a la fecha de ingreso';
        END IF;
        -- no hago join con empleado porque ya pasó una validación que controla eso
        v_id_empleado := (SELECT id_persona FROM persona WHERE dni = p_dni);
        UPDATE mantenimiento_cama SET fecha_egreso = p_fecha,
                id_empleado = v_id_empleado, estado = p_estado
        WHERE id_cama = p_id_cama AND fecha_egreso IS NULL;
        IF p_estado = 'reparado' THEN
                UPDATE cama SET estado = 'OK' WHERE id_cama = p_id_cama;
        END IF;
        IF p_estado = 'Fuera de Servicio' THEN
           UPDATE cama SET estado = 'FUERA DE SERVICIO' WHERE id_cama = p_id_cama;
        END IF;
END; $BODY$ LANGUAGE plpgsql
```

b) Separo las funcionalidades, una función solo para el UPDATE en la tabla internación, otra función solo para el INSERT en la tabla factura y finalmente un PROCEDURE que llama a ambas funciones.

```
CREATE OR REPLACE FUNCTION internacion_alta(
        p_dni character varying, p_fecha date, p_hora time without time zone,
        p_costo numeric(10,2))
        RETURNS boolean AS $BODY$
DECLARE
        v_id_paciente integer;
BEGIN
        -- validaciones de parámetros
        IF p_dni IS NULL OR NOT EXISTS(SELECT * FROM internacion
                                        INNER JOIN persona ON id_persona = id_paciente
                                        WHERE dni = p_dni AND fecha_alta IS NULL) THEN
                -- RAISE EXCEPTION 'No existe un paciente internado bajo ese DNI';
                RETURN false;
        END IF;
        IF p_fecha IS NULL THEN
                -- RAISE EXCEPTION 'Debe ingresar una fecha de alta';
                RETURN false;
        END IF;
        IF p_costo IS NULL OR p_costo < 1 THEN
                -- RAISE EXCEPTION 'Debe ingresar un costo de internación';
                RETURN false;
        END IF;

        -- no hago join con paciente porque ya pasó una validación que controla eso
        v_id_paciente := (SELECT id_persona FROM persona WHERE dni = p_dni);

        UPDATE internacion SET fecha_alta = p_fecha, hora = p_hora, costo = p_costo
        WHERE id_paciente = v_id_paciente AND fecha_alta IS NULL;

        RETURN true;
END;
$BODY$ LANGUAGE plpgsql
```

Trabajo Práctico Nro. 7 - Ciclo 2023

```
CREATE OR REPLACE FUNCTION factura_alta(
        p_dni character varying, p_fecha date, p_hora time without time zone,
        p_monto numeric(10,2))
        RETURNS boolean AS $BODY$
DECLARE
        v_id_paciente integer;
BEGIN
        -- validaciones de parámetros
        IF p_dni IS NULL OR NOT EXISTS(SELECT * FROM paciente
                                        INNER JOIN persona ON id_persona = id_paciente
                                        WHERE dni = p_dni) THEN
                -- RAISE EXCEPTION 'No existe un paciente bajo ese DNI';
                RETURN false;
        END IF;
        IF p_fecha IS NULL THEN
                -- RAISE EXCEPTION 'Debe ingresar una fecha de alta';
                RETURN false;
        END IF;
        IF p_monto IS NULL OR p_monto < 1 THEN
                -- RAISE EXCEPTION 'Debe ingresar un monto';
                RETURN false;
        END IF;

        -- no hago join con paciente porque ya pasó una validación que controla eso
        v_id_paciente := (SELECT id_persona FROM persona WHERE dni = p_dni);

        INSERT INTO factura VALUES((SELECT MAX(id_factura) + 1 FROM factura),
                                v_id_paciente, p_fecha, p_hora, p_monto, 'N', p_monto);

        RETURN true;
END;
$BODY$ LANGUAGE plpgsql



CREATE OR REPLACE PROCEDURE internacion_factura_alta(
        IN p_dni character varying,
        IN p_fecha date,
        IN p_hora time without time zone,
        IN p_costo numeric(10,2) )
LANGUAGE 'plpgsql' AS $BODY$
DECLARE
        v_alta boolean;
        v_factura boolean;
```

```
BEGIN
        v_alta := (SELECT internacion_alta(p_dni, p_fecha, p_hora, p_costo));
        IF v_alta THEN
                v_factura := (SELECT factura_alta(p_dni, p_fecha, p_hora, p_costo));
                IF v_factura THEN
                        RAISE NOTICE 'Alta exitosa del paciente';
                ELSE
                        RAISE EXCEPTION 'No se pudo emitir la factura de internación';
                END IF;
        ELSE
                RAISE EXCEPTION 'No se pudo hacer el alta del paciente';
        END IF;
END; $BODY$
```

c) Separo las funcionalidades, una función solo para el INSERT en la tabla diagnóstico, otra función solo para el INSERT en la tabla tratamiento y finalmente un PROCEDURE que llama a ambas funciones.

```
CREATE OR REPLACE FUNCTION diagnostico_asignar(
        p_dni_paciente character varying, p_dni_medico character varying,
        p_fecha date, p_descripcion character varying, p_patologia character varying)
        RETURNS boolean AS $BODY$
DECLARE
        v_id_paciente integer;
        v_id_medico integer;
        v_id_patologia smallint;
BEGIN
        -- validaciones de parámetros
        IF p_dni_paciente IS NULL OR NOT EXISTS(SELECT * FROM paciente
                                        INNER JOIN persona ON id_persona = id_paciente
                                        WHERE dni = p_dni_paciente) THEN
                -- RAISE EXCEPTION 'No existe un paciente bajo ese DNI';
                RETURN false;
        END IF;
        IF p_dni_medico IS NULL OR NOT EXISTS(SELECT * FROM empleado
                                        INNER JOIN persona ON id_persona = id_empleado
                                        WHERE dni = p_dni_medico) THEN
                -- RAISE EXCEPTION 'No existe un médico bajo ese DNI';
                RETURN false;
        END IF;
        IF p_fecha IS NULL THEN
                -- RAISE EXCEPTION 'Debe ingresar una fecha de consulta';
                RETURN false;
        END IF;
```

```
        IF p_patologia IS NULL OR NOT EXISTS(SELECT * FROM patologia
                                                WHERE nombre = p_patologia) THEN
            -- RAISE EXCEPTION 'No existe la patologia ingresada';
            RETURN false;
        END IF;
        v_id_paciente := (SELECT id_persona FROM persona WHERE dni = p_dni_paciente);
        v_id_medico := (SELECT id_persona FROM persona WHERE dni = p_dni_medico);
        v_id_patologia := (SELECT id_patologia FROM patologia
                                WHERE nombre = p_patologia);
        IF NOT EXISTS(SELECT * FROM consulta
            WHERE id_paciente = v_id_paciente
            AND id_empleado = v_id_medico AND fecha = p_fecha) THEN
            -- RAISE EXCEPTION 'No existe una consulta con los parámetros ingresados';
            RETURN false;
        END IF;

        INSERT INTO diagnostico
        VALUES (v_id_paciente, v_id_medico, p_fecha, p_descripcion, v_id_patologia);
        RETURN true;
END;
$BODY$ LANGUAGE plpgsql

CREATE OR REPLACE FUNCTION tratamiento_alta(p_dni_paciente character varying,
        p_dni_medico character varying, p_fecha date, p_medicamento character varying,
        p_descripcion character varying, p_dosis character varying, p_costo numeric(10,2))
        RETURNS boolean AS $BODY$
DECLARE
        v_id_paciente integer;
        v_id_medico integer;
        v_id_medicamento integer;
BEGIN
        -- validaciones de parámetros
        IF p_dni_paciente IS NULL OR NOT EXISTS(SELECT * FROM paciente
                                        INNER JOIN persona ON id_persona = id_paciente
                                        WHERE dni = p_dni_paciente) THEN
            -- RAISE EXCEPTION 'No existe un paciente bajo ese DNI';
            RETURN false;
        END IF;
        IF p_dni_medico IS NULL OR NOT EXISTS(SELECT * FROM empleado
                                        INNER JOIN persona ON id_persona = id_empleado
                                        WHERE dni = p_dni_medico) THEN
            -- RAISE EXCEPTION 'No existe un médico bajo ese DNI';
            RETURN false;
        END IF;
```

```
        IF p_fecha IS NULL THEN
                -- RAISE EXCEPTION 'Debe ingresar una fecha de tratamiento';
                RETURN false;
        END IF;
        IF p_medicamento IS NULL OR NOT EXISTS(SELECT * FROM medicamento
                                                WHERE nombre = p_medicamento) THEN
                -- RAISE EXCEPTION 'No existe el medicamento ingresado';
                RETURN false;
        END IF;
        IF p_costo IS NULL OR p_costo < 1 THEN
                -- RAISE EXCEPTION 'Debe ingresar un costo';
                RETURN false;
        END IF;

        v_id_paciente := (SELECT id_persona FROM persona WHERE dni = p_dni_paciente);
        v_id_medico := (SELECT id_persona FROM persona WHERE dni = p_dni_medico);
        v_id_medicamento := (SELECT id_medicamento FROM medicamento
                                WHERE nombre = p_medicamento);

        INSERT INTO tratamiento VALUES(v_id_paciente, v_id_medicamento,
                p_fecha, v_id_medico, p_medicamento, p_descripcion, p_dosis, p_costo);

        RETURN true;
END;
$BODY$ LANGUAGE plpgsql



CREATE OR REPLACE PROCEDURE diagnostico_tratamiento_alta(
        IN p_dni_paciente character varying,
        IN p_dni_medico character varying,
        IN p_fecha date,
        IN p_desc_patologia character varying,
        IN p_patologia character varying,
        IN p_medicamento character varying,
        IN p_desc_tratamiento character varying,
        IN p_dosis character varying,
        IN p_costo numeric(10,2) )
LANGUAGE 'plpgsql' AS $BODY$
DECLARE
        v_diagnostico boolean;
        v_tratamiento boolean;
```

```
BEGIN
        v_diagnostico := (SELECT diagnostico_asignar(p_dni_paciente, p_dni_medico,
p_fecha, p_desc_patologia, p_patologia));
        IF v_diagnostico THEN
                v_tratamiento := (SELECT tratamiento_alta(p_dni_paciente, p_dni_medico,
p_fecha, p_medicamento, p_desc_tratamiento, p_dosis, p_costo));
                IF v_tratamiento THEN
                        RAISE NOTICE 'Alta exitosa del diagnóstico y tratamiento';
                ELSE
                        RAISE EXCEPTION 'No se pudo dar un tratamiento';
                END IF;
        ELSE
                RAISE EXCEPTION 'No se pudo asignar un diagnóstico';
        END IF;
END;
$BODY$
```