[2 PUNTOS] PROGRAMA 1

a) Crea el procedimiento pListarPoliticosDietas_porLetrayTipo

```
DROP PROCEDURE IF EXISTS pListarPoliticosDietas_porLetrayTipo;
DELIMITER $$
CREATE PROCEDURE pListarPoliticosDietas_porLetrayTipo (
	IN letraConcepto CHAR,
	IN tipoPolitico VARCHAR(2),
	OUT numResultados INTEGER)
BEGIN
	DECLARE countCA INTEGER;
	DECLARE countCO INTEGER;

	IF (tipoPolitico NOT IN('CA', 'CO'))
		THEN
			SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Parámetros inesperados\n*****\
n====> Mensaje de error.\n*****\n';
	END IF;

	CASE
		WHEN tipoPolitico = 'CA' THEN
			SELECT DISTINCT CA.dni_pol AS DNI, P.nombre AS Nombre, P.apellidos AS Apellidos
			FROM cargo CA, politico P, cobrar C
			WHERE CA.dni_pol = P.dni_pol AND C.dni_pol = CA.dni_pol AND
			C.concepto LIKE CONCAT(letraConcepto, '%')
			ORDER BY P.nombre, P.apellidos;

			SELECT COUNT(DISTINCT CA.dni_pol) INTO countCA
			FROM cargo CA, politico P, cobrar C
			WHERE CA.dni_pol = P.dni_pol AND C.dni_pol = CA.dni_pol AND
			C.concepto LIKE CONCAT(letraConcepto, '%')
			ORDER BY P.nombre, P.apellidos;

			SET numResultados = countCA;

		WHEN tipoPolitico = 'CO' THEN
			SELECT DISTINCT CO.dni_pol AS DNI, P.nombre AS Nombre, P.apellidos AS Apellidos
			FROM concejal CO, politico P, cobrar C
			WHERE CO.dni_pol = P.dni_pol AND C.dni_pol = CO.dni_pol AND
			C.concepto LIKE CONCAT(letraConcepto, '%')
			ORDER BY P.nombre, P.apellidos;

			SELECT COUNT(DISTINCT CO.dni_pol) INTO countCO
			FROM cargo CO, politico P, cobrar C
			WHERE CO.dni_pol = P.dni_pol AND C.dni_pol = CO.dni_pol AND
			C.concepto LIKE CONCAT(letraConcepto, '%')
			ORDER BY P.nombre, P.apellidos;

			SET numResultados = countCO;
```

```
        END CASE;
END$$
DELIMITER ;
```

[2 PUNTOS] PROGRAMA 2

a) Crea la función fNumCobrosDieta

```
DROP FUNCTION IF EXISTS fNumCobrosDieta;
DELIMITER $$
CREATE FUNCTION fNumCobrosDieta(
        conceptoEntrada VARCHAR(30))
RETURNS INTEGER
NOT DETERMINISTIC
READS SQL DATA
BEGIN
        DECLARE numeroCobros INTEGER;

        SELECT COUNT(C.dni_pol) INTO numeroCobros
        FROM cobrar C, dieta D
        WHERE D.concepto = C.concepto AND D.concepto = conceptoEntrada;

        RETURN numeroCobros;
END$$
DELIMITER ;
```

b) Crea la función fImporteCobrosDieta

```
DROP FUNCTION IF EXISTS fImporteCobrosDieta;
DELIMITER $$
CREATE FUNCTION fImporteCobrosDieta(
        conceptoEntrada VARCHAR(30))
RETURNS INTEGER
NOT DETERMINISTIC
READS SQL DATA
BEGIN
        DECLARE importeCobros INTEGER;

        SELECT SUM(C.importe) INTO importeCobros
        FROM cobrar C, dieta D
        WHERE D.concepto = C.concepto AND D.concepto = conceptoEntrada;

        RETURN importeCobros;
END$$
DELIMITER ;
```

[2 PUNTOS] PROGRAMA 3

a) Crea el trigger tComprobarFechaReunion

```
DROP TRIGGER IF EXISTS tComprobarFechaReunion;
DELIMITER $$
CREATE TRIGGER tComprobarFechaReunion
BEFORE INSERT ON reunion
FOR EACH ROW
BEGIN
	IF (NEW.fcelebracion <= NOW())
		THEN
			SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Operación no permitida\n*****\
n====> Mensaje de error.\n*****\n';
	END IF;
END$$
DELIMITER ;
```

** Se podría usar también CURDATE() en lugar de NOW(), si no queremos tener en cuenta la hora, además de la fecha (como ocurre con NOW()).*

[2 PUNTOS] PROGRAMA 4

a) Crea el trigger necesario para prevenir inserciones incorrectas en Cargo

```
DROP TRIGGER IF EXISTS tComprobarCargo;
DELIMITER $$
CREATE TRIGGER tComprobarCargo
BEFORE INSERT ON cargo
FOR EACH ROW
BEGIN
	IF EXISTS(SELECT 1 FROM concejal WHERE dni_pol = NEW.dni_pol)
		THEN
			SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Operación no permitida\n*****\
n====> Mensaje de error.\n*****\n';
	END IF;
END$$
DELIMITER ;
```

b) Crea el trigger necesario para prevenir inserciones incorrectas en Concejal

```
DROP TRIGGER IF EXISTS tComprobarConcejal;
DELIMITER $$
CREATE TRIGGER tComprobarConcejal
BEFORE INSERT ON concejal
FOR EACH ROW
BEGIN
	IF EXISTS(SELECT 1 FROM cargo WHERE dni_pol = NEW.dni_pol)
		THEN
			SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Operación no permitida\n*****\
n====> Mensaje de error.\n*****\n';
	END IF;
END$$
DELIMITER ;
```

[2 PUNTOS] PROGRAMA 5

a) Crea el trigger necesario para prevenir borrados de ASESOR

```
DROP TRIGGER IF EXISTS tBefore_delete_asesor;
DELIMITER $$
CREATE TRIGGER tBefore_delete_asesor
BEFORE DELETE ON asesor
FOR EACH ROW
BEGIN

	IF ((SELECT COUNT(*)
		FROM asesor
		WHERE dni_pol = OLD.dni_pol) = 1)

	THEN
		SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Operación no permitida\n*****\n====> Mensaje de error.\n*****\n';

	END IF;

END$$
DELIMITER ;
```

b) Crea el trigger necesario para prevenir actualizaciones del campo "dni_pol" de la tabla ASESOR

```
DROP TRIGGER IF EXISTS tBefore_update_asesor;
DELIMITER $$
CREATE TRIGGER tBefore_update_asesor
BEFORE UPDATE ON asesor
FOR EACH ROW
BEGIN

IF (OLD.dni_pol <> NEW.dni_pol)

AND
	((SELECT COUNT(*)
	FROM asesor
	WHERE dni_pol = OLD.dni_pol) = 1)

THEN
	SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Operación no permitida\n*****\n====> Mensaje de error.\n*****\n';

END IF;

END$$
DELIMITER ;
```