**Entregar un enlace a github al producto desarrollado en cada una de las fases del proyecto, y un archivo zip con el proyecto git.  
  
El Readme de Github debe incluir una descripción de la GENERACIÓN DE CÓDIGO DE COMPONENTES Y PROCEDIMIENTOS (CSI-2)**

|  |
| --- |
| CÓDIGO DE COMPONENTES |
| DROP TABLE IF EXISTS `Centro\_Medico`.Flags; CREATE TABLE IF NOT EXISTS `Centro\_Medico`.Flags(  `varName` VARCHAR(255) NOT NULL,  `value` BOOLEAN NOT NULL DEFAULT FALSE,  PRIMARY KEY (`varName`),  UNIQUE INDEX `varName\_UNIQUE` (`varName` ASC)  ); DROP TABLE IF EXISTS `Centro\_Medico`.Cliente\_Compra\_Servicios\_AtravesDe\_Empleados; DROP TABLE IF EXISTS `Centro\_Medico`.Centros\_Recibe\_AsistenciaExterior; DROP TABLE IF EXISTS `Centro\_Medico`.Empleado\_Trabaja\_Centro; DROP TABLE IF EXISTS `Centro\_Medico`.Servicios;  DROP TABLE IF EXISTS `Centro\_Medico`.TipoServicios; DROP TABLE IF EXISTS `Centro\_Medico`.Centros; DROP TABLE IF EXISTS `Centro\_Medico`.AsistenciaExterior;  DROP TABLE IF EXISTS `Centro\_Medico`.ServiciosExternos;  DROP TABLE IF EXISTS `Centro\_Medico`.EmpresasExternas; DROP TABLE IF EXISTS `Centro\_Medico`.Patologias; DROP TABLE IF EXISTS `Centro\_Medico`.Clientes; DROP TABLE IF EXISTS `Centro\_Medico`.EmpleadoSecretario; DROP TABLE IF EXISTS `Centro\_Medico`.EmpleadoMedico;  DROP TABLE IF EXISTS `Centro\_Medico`.MedicoEspecializacion; DROP TABLE IF EXISTS `Centro\_Medico`.EmpleadoPsicologo; DROP TABLE IF EXISTS `Centro\_Medico`.EmpleadoAdministrador; DROP TABLE IF EXISTS `Centro\_Medico`.Empleados;  *--------------------------------------------------------*  *-- Table `Centro\_Medico`.`AsistenciaExterior`*  *-- AUXILIAR TABLE `Centro\_Medico`.`ServiciosExternos`*  *-- -----------------------------------------------------*  CREATE TABLE IF NOT EXISTS `Centro\_Medico`.ServiciosExternos(  id INT NOT NULL AUTO\_INCREMENT,  tipo VARCHAR(55) NOT NULL,  PRIMARY KEY (`id`),  UNIQUE INDEX `tipo\_UNIQUE` (`tipo` ASC)  ); *--------------------------------------------------------*  *-- AUXILIAR TABLE `Centro\_Medico`.`EmpresasExternas`*  *-- -----------------------------------------------------*  CREATE TABLE IF NOT EXISTS `Centro\_Medico`.EmpresasExternas(  id INT NOT NULL AUTO\_INCREMENT,  empresa VARCHAR(55) NOT NULL,  PRIMARY KEY (`id`),  UNIQUE INDEX `empresa\_UNIQUE` (`empresa` ASC)  ); CREATE TABLE IF NOT EXISTS `Centro\_Medico`.AsistenciaExterior(*/\*Entidad : 'AsistenciaExterior'\*/*  tipo INT NOT NULL*/\*ENUM ('Limpieza','Informática')\*/*,  empresa INT NOT NULL*/\*VARCHAR(55)\*/*,  PRIMARY KEY (`tipo`, `empresa`),  CONSTRAINT `FK\_ServiciosExternos\_tipo\_\_AsistenciaExterior`  FOREIGN KEY (`tipo`)  REFERENCES `Centro\_Medico`.ServiciosExternos(`id`)  ON DELETE RESTRICT  ON UPDATE CASCADE,    CONSTRAINT `FK\_EmpresasExternas\_empresa\_\_AsistenciaExterior`  FOREIGN KEY (`empresa`)  REFERENCES `Centro\_Medico`.EmpresasExternas(`id`)  ON DELETE RESTRICT  ON UPDATE CASCADE ); *-- -----------------------------------------------------* *-- Table `Centro\_Medico`.`Centros`* *-- -----------------------------------------------------* CREATE TABLE IF NOT EXISTS `Centro\_Medico`.Centros(*/\*Entidad : 'Centro'\*/*  direccion VARCHAR(255) NOT NULL,  horarioInicio TIME NOT NULL,  horarioFin TIME NOT NULL,  PRIMARY KEY(`direccion`),  UNIQUE INDEX `direccion\_UNIQUE` (`direccion` ASC)  */\*,CONSTRAINT `CHECK\_Centros\_\_horarioInicio\_lowerThan\_horarioFin` CHECK ( TIMEDIFF(horarioFin,horarioInicio) > 0 )\*/* *-- No funca* );  *-- -----------------------------------------------------* *-- Table `Centro\_Medico`.`Centros\_Recibe\_AsistenciaExterior`* *-- -----------------------------------------------------* CREATE TABLE IF NOT EXISTS `Centro\_Medico`.Centros\_Recibe\_AsistenciaExterior(*/\*Relación : 'Recibe'\*/*  tipoAsistenciaExterior INT NOT NULL,  empresaAsistenciaExterior INT NOT NULL,  direccionCentro VARCHAR(255) NOT NULL,   fecha DATETIME NOT NULL,  coste INT UNSIGNED NOT NULL,  PRIMARY KEY (`tipoAsistenciaExterior`,`empresaAsistenciaExterior`,`direccionCentro`,`fecha`),    CONSTRAINT `FK\_AsistenciaExterior\_tipoAsistenciaExterior\_\_C\_Recibe\_A`  FOREIGN KEY (`tipoAsistenciaExterior`)  REFERENCES `Centro\_Medico`.AsistenciaExterior(`tipo`)  ON DELETE RESTRICT  ON UPDATE CASCADE,    CONSTRAINT `FK\_AsistenciaExterior\_empresaAsistenciaExterior\_\_C\_Recibe\_A`  FOREIGN KEY (`empresaAsistenciaExterior`)  REFERENCES `Centro\_Medico`.AsistenciaExterior(`empresa`)  ON DELETE RESTRICT  ON UPDATE CASCADE,    CONSTRAINT `FK\_Centro\_direccionCentro\_\_C\_Recibe\_A`  FOREIGN KEY (`direccionCentro`)  REFERENCES `Centro\_Medico`.Centros(`direccion`)  ON DELETE RESTRICT  ON UPDATE CASCADE ); *-- -----------------------------------------------------* *-- Table `Centro\_Medico`.`Servicios`* *-- -----------------------------------------------------*  *-- AUXILIAR TABLE `Centro\_Medico`.`TipoServicios`*  *-- -----------------------------------------------------*  CREATE TABLE IF NOT EXISTS `Centro\_Medico`.TipoServicios(  id INT NOT NULL AUTO\_INCREMENT,  tipo VARCHAR(70) NOT NULL,  PRIMARY KEY (`id`),  UNIQUE INDEX `tipo\_UNIQUE` (`tipo` ASC)  ); CREATE TABLE IF NOT EXISTS `Centro\_Medico`.Servicios(*/\*Entidad : 'Servicios'\*/*  direccionCentro VARCHAR(255) NOT NULL,  tipo INT NOT NULL,  precioBase FLOAT NOT NULL,  `%administrador` FLOAT UNSIGNED NOT NULL,*/\*FLOAT(2,2) is deprecated\*/*  `%secretario` FLOAT UNSIGNED NOT NULL,  `%psicologo` FLOAT UNSIGNED NOT NULL,  `%medico` FLOAT UNSIGNED NOT NULL,  PRIMARY KEY(`direccionCentro`,`tipo`),    CONSTRAINT `FK\_Centros\_direccionCentro\_\_Servicios`  FOREIGN KEY (`direccionCentro`)  REFERENCES `Centro\_Medico`.Centros(`direccion`)  ON DELETE RESTRICT  ON UPDATE CASCADE,    CONSTRAINT `FK\_TipoServicios\_tipo\_\_Servicios`  FOREIGN KEY (`tipo`)  REFERENCES `Centro\_Medico`.TipoServicios(`id`)  ON DELETE RESTRICT  ON UPDATE CASCADE ); *-- -----------------------------------------------------* *-- Table `Centro\_Medico`.`Clientes`* *-- -----------------------------------------------------* CREATE TABLE IF NOT EXISTS `Centro\_Medico`.Clientes(*/\*Entidad : 'Clientes'\*/*  nombre VARCHAR(255) NOT NULL,  dni CHAR(9) NOT NULL,  telefono VARCHAR(50) NOT NULL,  fechaNacimiento DATE NOT NULL,  PRIMARY KEY(`dni`),  UNIQUE INDEX `dni\_UNIQUE` (`dni` ASC) ); *-- -----------------------------------------------------* *-- Table `Centro\_Medico`.`Patalogia`* *-- -----------------------------------------------------* CREATE TABLE IF NOT EXISTS `Centro\_Medico`.Patologias(*/\*Entidad : 'Patologias'\*/*  nombre VARCHAR(255) NOT NULL,  dniCliente CHAR(9) NOT NULL,  penalizacion FLOAT DEFAULT 1 NOT NULL,  PRIMARY KEY(`dniCliente`,`nombre`),    CONSTRAINT `FK\_Clientes\_dni\_\_Patologias`  FOREIGN KEY (`dniCliente`)  REFERENCES `Centro\_Medico`.Clientes(`dni`)  ON DELETE RESTRICT  ON UPDATE CASCADE ); *-- -----------------------------------------------------* *-- Table `Centro\_Medico`.`Empleados`* *-- -----------------------------------------------------* CREATE TABLE IF NOT EXISTS `Centro\_Medico`.Empleados(*/\*Entidad : 'Empleados'\*/*  dni CHAR(9) NOT NULL,  fechaIni DATETIME NOT NULL,  fechaFin DATETIME NOT NULL,  PRIMARY KEY(`dni`,fechaIni),  UNIQUE INDEX `dni\_UNIQUE` (`dni` ASC) );  CREATE TABLE IF NOT EXISTS `Centro\_Medico`.EmpleadoSecretario(  dniEmpleado CHAR(9) NOT NULL,  PRIMARY KEY(`dniEmpleado`),  UNIQUE INDEX `dniEmpleado\_UNIQUE` (`dniEmpleado` ASC),    CONSTRAINT `FK\_Empleadoss\_dniEmpleado\_\_EmpleadoSecretario`  FOREIGN KEY (`dniEmpleado`)  REFERENCES `Centro\_Medico`.Empleados(`dni`)  ON DELETE RESTRICT  ON UPDATE CASCADE  );  *-- -----------------------------------------------------*  *-- Auxiliar Table `Centro\_Medico`.`MedicoEspecializacion`*  *-- -----------------------------------------------------*  CREATE TABLE IF NOT EXISTS `Centro\_Medico`.MedicoEspecializacion(  id INT NOT NULL AUTO\_INCREMENT,  especializacion VARCHAR(70) NOT NULL,  PRIMARY KEY (`id`),  UNIQUE INDEX `tipo\_UNIQUE` (`especializacion` ASC)  );   CREATE TABLE IF NOT EXISTS `Centro\_Medico`.EmpleadoMedico(  dniEmpleado CHAR(9) NOT NULL,  especializacion INT NOT NULL,  PRIMARY KEY(`dniEmpleado`),  UNIQUE INDEX `dniEmpleado\_UNIQUE` (`dniEmpleado` ASC),  CONSTRAINT `FK\_Empleados\_dniEmpleado\_\_EmpleadoMedico`  FOREIGN KEY (`dniEmpleado`)  REFERENCES `Centro\_Medico`.Empleados(`dni`)  ON DELETE RESTRICT  ON UPDATE CASCADE,   CONSTRAINT `FK\_MedicoEspecializacion\_especializacion\_\_EmpleadoMedico`  FOREIGN KEY (`especializacion`)  REFERENCES `Centro\_Medico`.MedicoEspecializacion(`id`)  ON DELETE RESTRICT  ON UPDATE CASCADE  );  CREATE TABLE IF NOT EXISTS `Centro\_Medico`.EmpleadoPsicologo(  dniEmpleado CHAR(9) NOT NULL,  PRIMARY KEY(`dniEmpleado`),  UNIQUE INDEX `dniEmpleado\_UNIQUE` (`dniEmpleado` ASC),    CONSTRAINT `FK\_Empleadoss\_dniEmpleado\_\_EmpleadoPsicologo`  FOREIGN KEY (`dniEmpleado`)  REFERENCES `Centro\_Medico`.Empleados(`dni`)  ON DELETE RESTRICT  ON UPDATE CASCADE  );   CREATE TABLE IF NOT EXISTS `Centro\_Medico`.EmpleadoAdministrador(  dniEmpleado CHAR(9) NOT NULL,  PRIMARY KEY(`dniEmpleado`),  UNIQUE INDEX `dniEmpleado\_UNIQUE` (`dniEmpleado` ASC),    CONSTRAINT `FK\_Empleados\_dniEmpleado\_\_EmpleadoAdministrador`  FOREIGN KEY (`dniEmpleado`)  REFERENCES `Centro\_Medico`.Empleados(`dni`)  ON DELETE RESTRICT  ON UPDATE CASCADE  ); *-- -----------------------------------------------------* *-- Table `Centro\_Medico`.`Cliente\_Compra\_Servicios\_AtravesDe\_Empleados`* *-- -----------------------------------------------------* CREATE TABLE IF NOT EXISTS `Centro\_Medico`.Cliente\_Compra\_Servicios\_AtravesDe\_Empleados(*/\*Relacion : 'Compra'\*/*  dniEmpleado CHAR(9) NOT NULL,  dniCliente CHAR(9) NOT NULL,  direccionCentro VARCHAR(255) NOT NULL,  tipoServicio INT NOT NULL,  fecha DATE NOT NULL,  fechaCad DATE NOT NULL,  `coste` FLOAT NOT NULL, */\*Atributo Calculado\*/*  PRIMARY KEY(`dniEmpleado`,`dniCliente`,`direccionCentro`,`tipoServicio`,`fecha`),    CONSTRAINT `FK\_Empleados\_dniEmpleado\_\_C\_Compra\_S\_AD\_E`  FOREIGN KEY (`dniEmpleado`)  REFERENCES `Centro\_Medico`.Empleados(`dni`)  ON DELETE RESTRICT  ON UPDATE CASCADE,    CONSTRAINT `FK\_Clientes\_dniCliente\_\_C\_Compra\_S\_AD\_E`  FOREIGN KEY (`dniCliente`)  REFERENCES `Centro\_Medico`.Clientes(`dni`)  ON DELETE RESTRICT  ON UPDATE CASCADE,    CONSTRAINT `FK\_Centros\_direccionCentro\_\_C\_Compra\_S\_AD\_E`  FOREIGN KEY (`direccionCentro`)  REFERENCES `Centro\_Medico`.Centros(`direccion`)  ON DELETE RESTRICT  ON UPDATE CASCADE,    CONSTRAINT `FK\_Servicios\_tipoServicio\_\_C\_Compra\_S\_AD\_E`  FOREIGN KEY (`tipoServicio`)  REFERENCES `Centro\_Medico`.Servicios(`tipo`)  ON DELETE RESTRICT  ON UPDATE CASCADE ); *-- -----------------------------------------------------* *-- Table `Centro\_Medico`.`Empleado\_Trabaja\_Centro`* *-- -----------------------------------------------------* CREATE TABLE IF NOT EXISTS `Centro\_Medico`.Empleado\_Trabaja\_Centro(*/\*Relación : 'Trabaja'\*/*  dniEmpleado CHAR(9) NOT NULL,  direccionCentro VARCHAR(255) NOT NULL,  tipoServicio INT NOT NULL,  fecha DATETIME NOT NULL,  `salario/día` FLOAT NULL, */\*Atributo Calculado\*/*  PRIMARY KEY(`dniEmpleado`,`direccionCentro`,`tipoServicio`, `fecha`),    CONSTRAINT `FK\_Empleados\_dniEmpleado\_\_E\_Trabaja\_C`  FOREIGN KEY (`dniEmpleado`)  REFERENCES `Centro\_Medico`.Empleados(`dni`)  ON DELETE RESTRICT  ON UPDATE CASCADE,    CONSTRAINT `FK\_Servicios\_direccionCentro\_\_E\_Trabaja\_C`  FOREIGN KEY (`direccionCentro`)  REFERENCES `Centro\_Medico`.Servicios(`direccionCentro`)  ON DELETE RESTRICT  ON UPDATE CASCADE,    CONSTRAINT `FK\_Servicios\_tipoServicio\_\_E\_Trabaja\_C`  FOREIGN KEY (`tipoServicio`)  REFERENCES `Centro\_Medico`.Servicios(`tipo`)  ON DELETE RESTRICT  ON UPDATE CASCADE ); |

|  |
| --- |
| CÓDIGO DE PROCEDIMIENTOS DE OPERACIÓN Y SEGURIDAD |
| DROP FUNCTION IF EXISTS `getCosteBaseServicio`; DELIMITER // CREATE FUNCTION getCosteBaseServicio(tipoServicio INT,direccionCentro VARCHAR(255)) RETURNS FLOAT BEGIN  DECLARE coste FLOAT;  SELECT Servicios.precioBase INTO coste from Servicios where Servicios.tipo = tipo and Servicios.direccionCentro = direccionCentro LIMIT 1;  RETURN coste; END; // DELIMITER ;   DROP FUNCTION IF EXISTS `getSumPenalizacionPatalogiasCliente`; DELIMITER // CREATE FUNCTION getSumPenalizacionPatalogiasCliente (dniCliente CHAR(9)) RETURNS FLOAT BEGIN  DECLARE sum FLOAT;  SELECT SUM(Patologias.penalizacion) INTO sum from Patologias where Patologias.dniCliente = dniCliente;  RETURN sum; END; // DELIMITER ;         DROP FUNCTION IF EXISTS `getEdadCliente`; DELIMITER // CREATE FUNCTION getEdadCliente (dniCliente CHAR(9)) RETURNS FLOAT BEGIN  DECLARE currentDate DATE;  DECLARE currentYear INT;  DECLARE currentMonth INT;  DECLARE bornYear INT;  DECLARE bornMonth INT;  DECLARE result INT;  SET currentDate = NOW();  SET currentYear = YEAR(currentDate);  SET currentMonth = MONTH(currentDate);  select YEAR(Clientes.fechaNacimiento) into bornYear from Clientes where Clientes.dni = dniCliente;  select MONTH(Clientes.fechaNacimiento) into bornMonth from Clientes where Clientes.dni = dniCliente;  SET result = currentYear - bornYear - ((currentMonth-bornMonth) < 0);  RETURN result; END; // DELIMITER ;               DROP FUNCTION IF EXISTS `calcCoste`; DELIMITER // CREATE FUNCTION calcCoste ( dniCliente CHAR(9),tipoServicio INT,direccionCentro VARCHAR(255), fechaIni DATE, fechaFin DATE) RETURNS FLOAT BEGIN  DECLARE penalizaciones FLOAT;  DECLARE clientAge INT;  DECLARE ageFactor FLOAT;  DECLARE daysDiff INT;  DECLARE costePerYear INT;  DECLARE costeBase FLOAT;  DECLARE costeFinal FLOAT;  SELECT getSumPenalizacionPatalogiasCliente(dniCliente) INTO penalizaciones;  SELECT getCosteBaseServicio(tipoServicio, direccionCentro) INTO costeBase;  SELECT getEdadCliente(dniCliente) INTO clientAge;  CASE   WHEN clientAge < 18 THEN SET ageFactor = 0.8;  WHEN clientAge >= 18 AND clientAge < 40 THEN SET ageFactor = 1;  WHEN clientAge >= 40 AND clientAge < 65 THEN SET ageFactor = 1.20;  WHEN clientAge > 65 THEN SET ageFactor = 1.40;  ELSE BEGIN END;  END CASE;    SET daysDiff = DATEDIFF(fechaFin, fechaIni);  CASE   WHEN (daysDiff < 365) THEN SET costePerYear = 20;  WHEN (daysDiff >= 365) AND daysDiff < 730 THEN SET costePerYear = 35;  WHEN daysDiff >= 730 THEN SET costePerYear = 40;  ELSE BEGIN END;  END CASE;   SET costeFinal = costeBase\*ageFactor + penalizaciones + costePerYear;  RETURN costeFinal; END; // DELIMITER ;   DROP FUNCTION IF EXISTS `thereIsMinPsicologoMedicoSecretario`; DELIMITER // CREATE FUNCTION thereIsMinPsicologoMedicoSecretario (direccion VARCHAR(255)) RETURNS BOOLEAN BEGIN  DECLARE isEnabled BOOLEAN;  DECLARE sumPsicologos INT;  DECLARE sumSecretarios INT;  DECLARE sumMedicos INT;  SELECT Flags.`value` INTO isEnabled FROM Flags where Flags.varName = 'EnableCheckMinPsicologoMedicoSecretario';  IF isEnabled = FALSE THEN  return TRUE;  END IF;  SELECT COUNT(ETC.dniEmpleado) INTO sumPsicologos   from Empleado\_Trabaja\_Centro as ETC   inner join EmpleadoPsicologo as EP on EP.dniEmpleado = ETC.dniEmpleado  inner join Empleados as E on E.dni = ETC.dniEmpleado  where   ETC.direccionCentro = direccion  AND  DATEDIFF(E.fechaFin, Now()) > 0;   SELECT COUNT(ETC.dniEmpleado) INTO sumSecretarios   from Empleado\_Trabaja\_Centro as ETC   inner join EmpleadoSecretario as ES on ES.dniEmpleado = ETC.dniEmpleado  inner join Empleados as E on E.dni = ETC.dniEmpleado  where   ETC.direccionCentro = direccion  AND  DATEDIFF(E.fechaFin, Now()) > 0;  SELECT COUNT(ETC.dniEmpleado) INTO sumMedicos   from Empleado\_Trabaja\_Centro as ETC   inner join EmpleadoMedico as EM on EM.dniEmpleado = ETC.dniEmpleado  inner join Empleados as E on E.dni = ETC.dniEmpleado  where   ETC.direccionCentro = direccion  AND  DATEDIFF(E.fechaFin, Now()) > 0;  RETURN ((sumPsicologos > 0) AND (sumSecretarios > 0) AND (sumMedicos > 0)); END; // DELIMITER ;  DROP FUNCTION IF EXISTS `IsEmpleadoSecretario`; DELIMITER // CREATE FUNCTION IsEmpleadoSecretario (dniEmpleado CHAR(9)) RETURNS FLOAT BEGIN  DECLARE exist BOOLEAN;  SELECT (COUNT(ES.dniEmpleado) > 0) INTO exist from EmpleadoSecretario as ES where ES.dniEmpleado = dniEmpleado;  RETURN exist; END; // DELIMITER ;   DROP FUNCTION IF EXISTS `IsEmpleadoMedico`; DELIMITER // CREATE FUNCTION IsEmpleadoMedico (dniEmpleado CHAR(9)) RETURNS FLOAT BEGIN  DECLARE exist BOOLEAN;  SELECT (COUNT(EM.dniEmpleado) > 0) INTO exist from EmpleadoMedico as EM where EM.dniEmpleado = dniEmpleado;  RETURN exist; END; // DELIMITER ;  DROP FUNCTION IF EXISTS `IsEmpleadoPsicologo`; DELIMITER // CREATE FUNCTION IsEmpleadoPsicologo (dniEmpleado CHAR(9)) RETURNS FLOAT BEGIN  DECLARE exist BOOLEAN;  SELECT (COUNT(EP.dniEmpleado) > 0) INTO exist from EmpleadoPsicologo as EP where EP.dniEmpleado = dniEmpleado;  RETURN exist; END; // DELIMITER ;  DROP FUNCTION IF EXISTS `IsEmpleadoAdministrador`; DELIMITER // CREATE FUNCTION IsEmpleadoAdministrador (dniEmpleado CHAR(9)) RETURNS FLOAT BEGIN  DECLARE exist BOOLEAN;  SELECT (COUNT(EA.dniEmpleado) > 0) INTO exist from EmpleadoAdministrador as EA where EA.dniEmpleado = dniEmpleado;  RETURN exist; END; // DELIMITER ;  DROP FUNCTION IF EXISTS `calcSalary`; DELIMITER // CREATE FUNCTION calcSalary (dniEmpleado CHAR(9)) RETURNS FLOAT BEGIN  DECLARE sum FLOAT;  DECLARE temp FLOAT;  SET sum := 0;  IF IsEmpleadoSecretario(dniEmpleado) = TRUE THEN  SELECT SUM(C\_C\_S\_E.Coste\*S.`%secretario`) INTO temp FROM Servicios AS S  INNER JOIN Cliente\_Compra\_Servicios\_AtravesDe\_Empleados AS C\_C\_S\_E ON S.tipo = C\_C\_S\_E.tipoServicio  INNER JOIN Empleados AS E ON E.dni = C\_C\_S\_E.dniEmpleado  WHERE E.dni = dniEmpleado;  SET sum := sum + temp;  END IF;   IF IsEmpleadoMedico(dniEmpleado) = TRUE THEN  SELECT SUM(C\_C\_S\_E.Coste\*S.`%medico`) INTO temp FROM Servicios AS S  INNER JOIN Cliente\_Compra\_Servicios\_AtravesDe\_Empleados AS C\_C\_S\_E ON S.tipo = C\_C\_S\_E.tipoServicio  INNER JOIN Empleados AS E ON E.dni = C\_C\_S\_E.dniEmpleado  WHERE E.dni = dniEmpleado;  SET sum := sum + temp;  END IF;  IF IsEmpleadoPsicologo(dniEmpleado) = TRUE THEN  SELECT SUM(C\_C\_S\_E.Coste\*S.`%psicologo`) INTO temp FROM Servicios AS S  INNER JOIN Cliente\_Compra\_Servicios\_AtravesDe\_Empleados AS C\_C\_S\_E ON S.tipo = C\_C\_S\_E.tipoServicio  INNER JOIN Empleados AS E ON E.dni = C\_C\_S\_E.dniEmpleado  WHERE E.dni = dniEmpleado;  SET sum := sum + temp;  END IF;  IF IsEmpleadoAdministrador(dniEmpleado) = TRUE THEN  SELECT SUM(C\_C\_S\_E.Coste\*S.`%administrador`) INTO temp FROM Servicios AS S  INNER JOIN Cliente\_Compra\_Servicios\_AtravesDe\_Empleados AS C\_C\_S\_E ON S.tipo = C\_C\_S\_E.tipoServicio  INNER JOIN Empleados AS E ON E.dni = C\_C\_S\_E.dniEmpleado  WHERE E.dni = dniEmpleado;   SET sum := sum + temp;  END IF;  return sum; END; // DELIMITER ;  DROP FUNCTION IF EXISTS `isInOtherEmpleadoTable`;  DELIMITER // CREATE FUNCTION isInOtherEmpleadoTable (dniEmpleado CHAR(9), excludeTable INT) RETURNS BOOLEAN BEGIN  DECLARE sum INT;  DECLARE temp INT;  SET sum := 0;  CASE excludeTable  WHEN 1 THEN *-- 'PSICOLOGO'*  SELECT COUNT(E.dniEmpleado) INTO temp FROM EmpleadoMedico as E WHERE E.dniEmpleado = dniEmpleado ;  SET sum := sum + temp;  SELECT COUNT(E.dniEmpleado) INTO temp FROM EmpleadoAdministrador as E WHERE E.dniEmpleado = dniEmpleado ;  SET sum := sum + temp;  SELECT COUNT(E.dniEmpleado) INTO temp FROM EmpleadoSecretario as E WHERE E.dniEmpleado = dniEmpleado ;  SET sum := sum + temp;  WHEN 2 THEN *-- 'MEDICO'*  SELECT COUNT(E.dniEmpleado) INTO temp FROM EmpleadoPsicologo as E WHERE E.dniEmpleado = dniEmpleado ;  SET sum := sum + temp;  SELECT COUNT(E.dniEmpleado) INTO temp FROM EmpleadoAdministrador as E WHERE E.dniEmpleado = dniEmpleado ;  SET sum := sum + temp;  SELECT COUNT(E.dniEmpleado) INTO temp FROM EmpleadoSecretario as E WHERE E.dniEmpleado = dniEmpleado ;  SET sum := sum + temp;  WHEN 3 THEN *-- 'SECRETARIO'*  SELECT COUNT(E.dniEmpleado) INTO temp FROM EmpleadoMedico as E WHERE E.dniEmpleado = dniEmpleado ;  SET sum := sum + temp;  SELECT COUNT(E.dniEmpleado) INTO temp FROM EmpleadoAdministrador as E WHERE E.dniEmpleado = dniEmpleado ;  SET sum := sum + temp;  SELECT COUNT(E.dniEmpleado) INTO temp FROM EmpleadoPsicologo as E WHERE E.dniEmpleado = dniEmpleado ;  SET sum := sum + temp;  WHEN 4 THEN *-- 'ADMINISTRADOR'*  SELECT COUNT(E.dniEmpleado) INTO temp FROM EmpleadoMedico as E WHERE E.dniEmpleado = dniEmpleado ;  SET sum := sum + temp;  SELECT COUNT(E.dniEmpleado) INTO temp FROM EmpleadoAdministrador as E WHERE E.dniEmpleado = dniEmpleado ;  SET sum := sum + temp;  SELECT COUNT(E.dniEmpleado) INTO temp FROM EmpleadoSecretario as E WHERE E.dniEmpleado = dniEmpleado ;  SET sum := sum + temp;  ELSE BEGIN END;  END CASE;  RETURN (sum  .0> 0); END; // DELIMITER ;  DELIMITER // CREATE TRIGGER `TRIGGER\_ServiciosExternos\_BI\_toUpperTipo` BEFORE INSERT on `Centro\_Medico`.ServiciosExternos FOR EACH ROW BEGIN  set NEW.tipo := UPPER(NEW.tipo); END// DELIMITER ;  DELIMITER // CREATE TRIGGER `TRIGGER\_EmpresasExternas\_BI\_toUpperEmpresa` BEFORE INSERT on `Centro\_Medico`.EmpresasExternas FOR EACH ROW BEGIN  set NEW.empresa := UPPER(NEW.empresa); END// DELIMITER ;  DELIMITER // CREATE TRIGGER `TRIGGER\_Centros\_BI\_toUpperDireccion\_And\_hInicio\_lt\_hFin` BEFORE INSERT on `Centro\_Medico`.Centros FOR EACH ROW BEGIN  set NEW.direccion := UPPER(NEW.direccion);   IF TIMEDIFF(NEW.horarioFin,NEW.horarioInicio) < 0 THEN  signal sqlstate '20002' set message\_text = 'Centros: El horarioFin debe ser mayor que el horarioInicio seteado';  END IF; END// DELIMITER ;  DELIMITER // CREATE TRIGGER `TRIGGER\_TipoServicios\_BI\_toUpperTipo` BEFORE INSERT on `Centro\_Medico`.TipoServicios FOR EACH ROW BEGIN  set NEW.tipo := UPPER(NEW.tipo); END// DELIMITER ;  DELIMITER // CREATE TRIGGER `TRIGGER\_Servicios\_BI\_toUpperDireccionAndSumEQ1` BEFORE INSERT on `Centro\_Medico`.Servicios FOR EACH ROW BEGIN  DECLARE sum INT;  set NEW.direccionCentro := UPPER(NEW.direccionCentro);  set sum := NEW.`%administrador`+ NEW.`%secretario`+ NEW.`%psicologo`+ NEW.`%medico`;  IF sum <> 1 THEN  signal sqlstate '20001' set message\_text = 'Servicios: El sumatorio de los porcentajes debe ser 1';  END IF; END// DELIMITER ; DELIMITER // CREATE TRIGGER `TRIGGER\_Clientes\_BI\_toUpper` BEFORE INSERT on `Centro\_Medico`.Clientes FOR EACH ROW BEGIN  set NEW.nombre := UPPER(NEW.nombre);  set NEW.dni := UPPER(NEW.dni); END// DELIMITER ;  DELIMITER // CREATE TRIGGER `TRIGGER\_Patologias\_BI\_toUpper` BEFORE INSERT on `Centro\_Medico`.Patologias FOR EACH ROW BEGIN  set NEW.nombre := UPPER(NEW.nombre);  set NEW.dniCliente := UPPER(NEW.dniCliente); END// DELIMITER ;  DELIMITER // CREATE TRIGGER `TRIGGER\_Empleados\_BI\_toUpper\_And\_fIni\_lt\_fFin` BEFORE INSERT on `Centro\_Medico`.Empleados FOR EACH ROW BEGIN  set NEW.dni := UPPER(NEW.dni);  IF (TIMESTAMP(NEW.fechaFin) - TIMESTAMP(NEW.fechaIni)) < 0 THEN  signal sqlstate '20003' set message\_text = 'Empleados : La fechaFin debe ser mayor que la fechaIni seteada';  END IF; END// DELIMITER ;  DELIMITER // CREATE TRIGGER `TRIGGER\_EmpleadoSecretario\_BI\_NotInOtherEmpleadoTable` BEFORE INSERT on `Centro\_Medico`.EmpleadoSecretario FOR EACH ROW BEGIN  set NEW.dniEmpleado := UPPER(NEW.dniEmpleado);  IF isInOtherEmpleadoTable(NEW.dniEmpleado , 3) THEN  signal sqlstate '20006' set message\_text = 'EmpleadoSecretario :El empleado ya se encuentra en otra tabla subclase de Empleados';  END IF; END// DELIMITER ;  DELIMITER // CREATE TRIGGER `TRIGGER\_EmpleadoSecretario\_BU\_NotInOtherEmpleadoTable` BEFORE UPDATE on `Centro\_Medico`.EmpleadoSecretario FOR EACH ROW BEGIN  set NEW.dniEmpleado := UPPER(NEW.dniEmpleado);  IF isInOtherEmpleadoTable(NEW.dniEmpleado , 3) THEN  signal sqlstate '20006' set message\_text = 'EmpleadoSecretario :El empleado ya se encuentra en otra tabla subclase de Empleados';  END IF; END// DELIMITER ;  DELIMITER // CREATE TRIGGER `TRIGGER\_EmpleadoMedico\_BI\_NotInOtherEmpleadoTable` BEFORE INSERT on `Centro\_Medico`.EmpleadoMedico FOR EACH ROW BEGIN  set NEW.dniEmpleado := UPPER(NEW.dniEmpleado);  IF isInOtherEmpleadoTable(NEW.dniEmpleado , 2) THEN  signal sqlstate '20006' set message\_text = 'EmpleadoMedico :El empleado ya se encuentra en otra tabla subclase de Empleados';  END IF; END// DELIMITER ;  DELIMITER // CREATE TRIGGER `TRIGGER\_EmpleadoMedico\_BU\_NotInOtherEmpleadoTable` BEFORE UPDATE on `Centro\_Medico`.EmpleadoMedico FOR EACH ROW  BEGIN  set NEW.dniEmpleado := UPPER(NEW.dniEmpleado);  IF isInOtherEmpleadoTable(NEW.dniEmpleado , 2) THEN  signal sqlstate '20006' set message\_text = 'EmpleadoSecretario :El empleado ya se encuentra en otra tabla subclase de Empleados';  END IF;  END// DELIMITER ; DELIMITER // CREATE TRIGGER `TRIGGER\_EmpleadoPsicologo\_BI\_NotInOtherEmpleadoTable` BEFORE INSERT on `Centro\_Medico`.EmpleadoPsicologo FOR EACH ROW  BEGIN  set NEW.dniEmpleado := UPPER(NEW.dniEmpleado);  IF isInOtherEmpleadoTable(NEW.dniEmpleado , 1) THEN  signal sqlstate '20006' set message\_text = 'EmpleadoPsicologo :El empleado ya se encuentra en otra tabla subclase de Empleados';  END IF;  END// DELIMITER ;  DELIMITER // CREATE TRIGGER `TRIGGER\_EmpleadoPsicologo\_BU\_NotInOtherEmpleadoTable` BEFORE UPDATE on `Centro\_Medico`.EmpleadoPsicologo FOR EACH ROW BEGIN  set NEW.dniEmpleado := UPPER(NEW.dniEmpleado);  IF isInOtherEmpleadoTable(NEW.dniEmpleado , 1) THEN  signal sqlstate '20006' set message\_text = 'EmpleadoPsicologo :El empleado ya se encuentra en otra tabla subclase de Empleados';  END IF; END// DELIMITER ;  DELIMITER // CREATE TRIGGER `TRIGGER\_EmpleadoAdministrador\_BI\_NotInOtherEmpleadoTable` BEFORE INSERT on `Centro\_Medico`.EmpleadoAdministrador FOR EACH ROW BEGIN  set NEW.dniEmpleado := UPPER(NEW.dniEmpleado);  IF isInOtherEmpleadoTable(NEW.dniEmpleado , 4) THEN  signal sqlstate '20006' set message\_text = 'EmpleadoAdministrador :El empleado ya se encuentra en otra tabla subclase de Empleados';  END IF; END// DELIMITER ;  DELIMITER // CREATE TRIGGER `TRIGGER\_EmpleadoAdministrador\_BU\_NotInOtherEmpleadoTable` BEFORE UPDATE on `Centro\_Medico`.EmpleadoAdministrador FOR EACH ROW  BEGIN  set NEW.dniEmpleado := UPPER(NEW.dniEmpleado);  IF isInOtherEmpleadoTable(NEW.dniEmpleado , 4) THEN  signal sqlstate '20006' set message\_text = 'EmpleadoAdministrador :El empleado ya se encuentra en otra tabla subclase de Empleados';  END IF; END// DELIMITER ;  DELIMITER // CREATE TRIGGER `TRIGGER\_C\_Compra\_S\_AD\_E\_BI\_toUpper\_AND\_f\_lt\_fCad\_AND\_coste\_calc` BEFORE INSERT on `Centro\_Medico`.Cliente\_Compra\_Servicios\_AtravesDe\_Empleados FOR EACH ROW BEGIN  set NEW.dniEmpleado := UPPER(NEW.dniEmpleado);  set NEW.dniCliente := UPPER(NEW.dniCliente);  set NEW.direccionCentro := UPPER(NEW.direccionCentro);  IF (TIMESTAMP(NEW.fechaCad) - TIMESTAMP(NEW.fecha)) < 0 THEN  signal sqlstate '20004' set message\_text = 'Cliente\_Compra\_Servicios\_AtravesDe\_Empleados : La fechaCad debe ser mayor que la fecha seteada';  END IF;  set NEW.coste := calcCoste(NEW.dniCliente,NEW.tipoServicio,NEW.direccionCentro,NEW.fecha,NEW.fechaCad); END// DELIMITER ;  DELIMITER // CREATE TRIGGER `TRIGGER\_E\_Trabaja\_C\_BI\_check` BEFORE INSERT on `Centro\_Medico`.Empleado\_Trabaja\_Centro FOR EACH ROW BEGIN  IF thereIsMinPsicologoMedicoSecretario(NEW.direccionCentro) = FALSE THEN  signal sqlstate '20005' set message\_text = 'Empleado\_Trabaja\_Centro: Num. min. profesionals failing.Fix it disabling 1º the flag <EnableCheckMinPsicologoMedicoSecretario>'; END IF;  END// DELIMITER ;   DELIMITER // CREATE TRIGGER `TRIGGER\_E\_Trabaja\_C\_BD\_check` BEFORE DELETE on `Centro\_Medico`.Empleado\_Trabaja\_Centro FOR EACH ROW BEGIN  IF thereIsMinPsicologoMedicoSecretario(OLD.direccionCentro) = FALSE THEN  signal sqlstate '20005' set message\_text = 'Empleado\_Trabaja\_Centro: Num. min. profesionals failing.Fix it disabling 1º the flag <EnableCheckMinPsicologoMedicoSecretario>';  END IF; END// DELIMITER ;   DELIMITER // CREATE TRIGGER `TRIGGER\_E\_Trabaja\_C\_BU\_check` BEFORE UPDATE on `Centro\_Medico`.Empleado\_Trabaja\_Centro FOR EACH ROW BEGIN  IF thereIsMinPsicologoMedicoSecretario(NEW.direccionCentro) = FALSE THEN  signal sqlstate '20005' set message\_text = 'Empleado\_Trabaja\_Centro: Num. min. profesionals failing.Fix it disabling 1º the flag <EnableCheckMinPsicologoMedicoSecretario>';  END IF; END// DELIMITER ; |