**LINK A LA IMAGEN DEL PARCIAL**

**1)a)**

```
ALTER TABLE Problema ADD CONSTRAINT FK2
FOREIGN KEY (id_producto, id_sub_prod)
REFERENCES Sub_Prod (id_producto, id_sub_prod)
ON DELETE CASCADE ON UPDATE RESTRICT
```

//creo, si no me equivoco, que están al revés el delet/update con el de abajo (sería este C,C y el FK1 C,R)

```
ALTER TABLE Sub_Prod  ADD CONSTRAINT FK1
FOREIGN KEY (id_producto)
REFERENCES Producto (id_producto)
ON DELETE CASCADE ON UPDATE CASCADE
```

**b)**

```
ALTER TABLE Problema ADD CONSTRAINT FK_Reporta_Desarrollador
FOREIGN KEY (id_des_reporta)
REFERENCES Desarrollador (id_desar)
ON DELETE SET DEFAULT ON UPDATE RESTRICT

ALTER TABLE Problema ADD CONSTRAINT FK_aCargo_Desarrollador
FOREIGN KEY (id_des_a_cargo)
REFERENCES Desarrollador (id_desar)
ON DELETE SET NULL ON UPDATE RESTRICT
```

**c)** Puesto en el UPDATE del b)

**d)** La mayoria de los restrict puestos en los incisos anteriores, faltan estos:

```
ALTER TABLE Problema ADD CONSTRAINT FK_Reporta_Equipo
FOREIGN KEY (id_equ_reporta)
REFERENCES Desarrollador (id_equipo)
ON DELETE RESTRICT ON UPDATE RESTRICT

ALTER TABLE Problema ADD CONSTRAINT FK_aCargo_Equipo
FOREIGN KEY (id_equ_a_cargo)
REFERENCES Desarrollador (id_equipo)
ON DELETE RESTRICT ON UPDATE RESTRICT
```

//las FK van por pares, o sea (id_equ_reporta, id_des_reporta) y en la referencia (id_equipo, id_desar) que es lo que dijeron que fue un error común que les faltara una

**2.1)** La correcta es la b) ya que el ON DELETE esta en modo CASCADE en la declaración de la FOREIGN KEY.

**2.2)** a) //No se acepta por dos motivos: 1) porque el valor por defecto se debe aplicar al par FK. //2) creo que como en el ejemplo el id_equipo no se corresponde con ningún id_eq_a_cargo no se //debería poner null

b) Se acepta, ya que en la PARTIAL coinciden ambas, y en la SIMPLE hay al menos un null

c) Se aceptan por lo mismo que el punto anterior

d) No se acepta porque la FULL de la FK_A_Cargo no coinciden ambas

**3.a)**
```
SELECT D.*
FROM Desarrollador D, Problema P
WHERE P.id_des_reporta = P.id_des_a_cargo
```

**b)**
```
SELECT DISTINCT id_producto, COUNT(id_problema)
FROM Problema
WHERE date_part('year', fecha_reporte) = 2014
GROUP BY id_producto
```

**c)**
```
select des.*
 from dearrollador des
 where not exists (
        select *
         from producto inner join sub_prod
         where not exists {
                        select *
                        from problema
                        where problema.des_id = des.des_id
                        and producto.sub_prod_id = sub_prod.sub_prod_id
                )
)
```

**4.a)**
```
CREATE ASSERTION desarrollador_equipo
CHECK(
        NOT EXISTS(
                SELECT * FROM Desarrollador D, Problema P
                WHERE D.id_desar = P.id_des_reporta AND D.id_equipo != P.id_equ_a_cargo
        )
)

o

CHECK( id_equ_a_cargo = id_equ_reporta)
```

**b)**
```
CHECK( id_des_a_cargo != id_des_reporta)
```

**c)**
```
CREATE ASSERTION fecha_de_reporte
CHECK(
```

```
        NOT EXISTS(
            SELECT * FROM Desarrollador D, Problema P
            WHERE D.fecha_ingreso > P.fecha_reporte
            AND P.id_des_a_cargo = D.id_desar AND P.id_equ_a_cargo = D.id_equipo
        )
    )
```

**5.a)**
```
CREATE TRIGGER desarrollador_equipo
BEFORE INSERT OR UPDATE id_des_reporta, id_equ_a_cargo ON Problema
FOR EACH ROW EXECUTE PROCEDURE comprobar_des_equ();

CREATE TRIGGER fecha_de_reporte
BEFORE INSERT OR UPDATE fecha_reporte ON Problema
```

**b)**

**6.a)**
```
CREATE VIEW VISTA_PROBLEMA
AS SELECT * FROM PROBLEMA
WITH CHECK OPTION
```

**b)** **//**
```
UPDATE ej_6_a SET fecha_reporte = toDate('DD/MM/YYYY','25/05/2000')
WHERE id_problema=1;
```

**c)**
```
//CREATE VIEW  ej_6_c AS
SELECT id_equipo, count(*) FROM Problema JOIN Desarrollador ON (algo)
```

**d)** **//**por ejemplo la vista anterior no es actualizable, porque no mantiene la clave de alguna de las tablas, y tiene una función de agregación

**e)** **//**CREATE Trigger INSTEAD OF ej_6_c ...

**7.a)** Esquema de ejecución que resulta en una actualización perdida:
<T1, S1> , <T2, S3> . <T1, S2> , <T2, S4>

**b)** Las posibles ejecuciones seriales son:
<T1, S1> , <T1, S2> . <T2. S3> , <T2. S4>
<T2. S3> , <T2. S4> , <T1, S1> , <T1, S2>
Y sus resultados serían salario = 6892 y salario = 6881.6 respectivamente.

**c)** El salario sería 2992 ya que no se podrá realizar el commit de la transacción que no fallo de T1 y solo se guardaran los resultados de la transacción T2.

**8.a)** i) GRANT, REVOKE. Son las principales utilizadas para otorgar y remover privilegios en una BD.

**b)** i) Ud. puede ver el contenido de V1. Sin importar si no tengo permisos sobre la tabla base, aun así puedo ver el contenido de la vista de la cual me otorgaron permisos.

**c)** iv) Ninguna de las anteriores. WITH GRANT OPTION se utiliza para ceder únicamente permisos que son propios a otro usuario.