Utente(id, nome, cognome, livello\_corrente, episodio\_corrente)

Episodio(id, nome, numero)

Livello(numero, idepisodio, tipo)

Amicizia(idutente, idutente)

LivelliUtente(numeroepisodio, idutente, datasuperamento, punteggio)

1. Risolvere le seguenti query in algebra relazionale:
   1. Trovare gli utenti che non hanno superato nessun livello con punteggio superiore a 1000 [4 punti];   
        
      ϭpunteggio > 1000 (LivelliUtente ⋈idutente = id Utente)
   2. Trovare l’utente che ha giocato in tutti gli episodi [4 punti];

πid, idepisodio (Utente ⋈ δnumeroepisodio->idepisodio(LivelloUtente)) ÷ πidepisodio(Livello)

* 1. Per ogni episodio trovare gli utenti che hanno raggiunto il punteggio massimo [4 punti];   
     LivelliUtente1 = δnumeroepisodio->numeroepisodio1, idutente->idutente1,  
      datasuperamento->datasuperamento1, punteggio->punteggio1πidutente(LivelliUtente) – πidutente(ϭ punteggio<punteggio1(LivelliUtente ⋈idutente>idutente1 LivelliUtente1))

1. Risolvere le seguenti query in SQL:
   1. Trovare gli utenti che hanno superato un numero di livelli maggiore a quello medio calcolato su tutti gli utenti [4 punti];   
      CREATE VIEW NLivelliSuperati AS  
      SELECT COUNT(\*) as nlivellisup, idutente  
      FROM LivelliUtente  
      GROUP BY idutente   
        
      SELECT \*

FROM NLivelliSuperati  
WHERE nlivellisup >= ( SELECT avg(nlivellisup) FROM NLivelliSuperati )

* 1. Tra gli episodi con il numero massimo di livelli di tipo “A tempo”, trovare quelli superati da almeno due utenti [5 punti];   
       
     SELECT   
     FROM EPISODIO E JOIN LIVELLO L ON idepisodio = id  
     WHERE L.tipo = “A tempo” AND EXISTS  
     (  
     SELECT \*  
     FROM LivelliUtente LU1 JOIN LIVELLIUTENTE LU2 ON LU1.ne = LU2.ne  
     WHERE LU1.numeroepisodio = E.idepisodio AND LU1.idutente <> LU2.idutente  
     GROUP BY LU1.idutente  
     )

4. Implementare un vincolo che imponga che il livello corrente dell’utente deve appartenere al corrispondente episodio [4 punti];   
  
ALTER TABLE Utente  
ADD CONSTRAINT C1  
CHECK ( livellocorrente IN   
(  
SELECT numero  
FROM EPISODIO E JOIN LIVELLO L ON E.id = L.idepisodio  
WHERE E.id = episodiocorrente  
)

5. Implementare un trigger che dopo l’inserimento del superamento del livello nella tabella LivelliUtente aggiorna opportunamente gli attributi livello\_corrente e epsiodio\_corrente nella tabella Utente [4 punti].

CREATE TRIGGER t1 AFTER INSERT ON LivelliUtente  
FOR EACH ROW  
BEGIN  
 DECLARE numlivtot number;   
 DECLARE numlivsup number;  
  
 SELECT numero INTO numlivtot  
 FROM EPISODIO E  
 WHERE E.id = new.numeroepisodio;  
  
 SELECT COUNT(\*) INTO numlivsup   
 FROM LivelliUtente LU  
 WHERE LU.numeroepisodio = new.numeroepisodio AND LU.idutente = new.idutente;  
  
 if numlivsup >= numlivtot then  
 UPDATE Utente SET livellocorrente = 0, episodiocorrente = new.idepisodio +1  
 else  
 UPDATE Utente SET livellocorrente = new.livellocorrente + 1  
 endif;  
  
END