---Projet SQL Promo IASD4 de ZANUTTO Sara & KACHKACHI Slim

----------***SCHÉMA DE LA BASE DE DONNÉES***----------

---------création table des séries ----------
---------unicité sur le nom de la série /NB unicité série au sein d'une même PF non spécifiée ----
---------->une même série peut être diffusée par plusieurs PFs ----------
---------->unicité du nom de la série quel que soit la PF ----------
--------nom de la série et nom de la PF doivent être saisis—----------

```
CREATE TABLE Serie(
   SID serial,
   nom varchar(25) NOT NULL,
   plateforme varchar(50) NOT NULL,
   noteMoyenne float DEFAULT -1,
   NbNotes integer DEFAULT 0,
   CONSTRAINT PK_Serie PRIMARY KEY (SID),
   CONSTRAINT CK_UnicitéNomSerie UNIQUE (nom)
);
```

-------création table des personnes ----------
-------unicité du pseudo, mais aucune demandes sur l'unicité nom et/ou prénom ----------
------>on laisse la possibilité d'homonymes ----------
-------nom, prénom et pseudo doivent être saisis ----------

```
CREATE TABLE Personne(
   PID serial,
   nom varchar(25) NOT NULL,
   prenom varchar(25) NOT NULL,
   pseudo varchar(25) NOT NULL,
   CONSTRAINT PK_Personne PRIMARY KEY (PID),
  CONSTRAINT CK_UnicitéPseudo UNIQUE (pseudo)
);
```

-------création table des abonnements ----------
-------une même personne peut avoir plusieurs abonnements à une même ou non PF ----------
-------la suppression d'une personne entraîne la suppression de tous ses abonnement ---------

```
CREATE TABLE Abonnes(
   AID serial,
   PID integer NOT NULL,
   plateforme varchar(50) NOT NULL,
   CONSTRAINT PK_Serial PRIMARY KEY (AID),
   CONSTRAINT FK_abonnes FOREIGN KEY (PID) REFERENCES Personne(PID) ON DELETE CASCADE
);
```

------création table des évaluations ----------
------pas de lien entre tables Evaluation et Abonnés ----------
------>pas d'obligation à être abonné pour noter une série ----------
------on comprend de la consigne qu'une personne ne peut noter qu'une seule fois une série ------ ---------
quelque soit la PF ----------
------la suppression d'une personne entraîne la suppression de toutes ses évaluations----------
------aucune spécification sur la suppression d'une série => pas d'action ----------
------> passera à null si nécessaire ----------
------note quand saisie doit être entre 0 inclus et 5 inclus----------

```
CREATE TABLE Evaluation(
   EID serial,
   PID integer NOT NULL,
```

```
   SID integer NOT NULL,
   note integer default 5 ,
   CONSTRAINT PK_Eval PRIMARY KEY (EID),
   CONSTRAINT FK_Personne FOREIGN KEY (PID) REFERENCES Personne (PID) ON DELETE CASCADE,
   CONSTRAINT FK_Serie FOREIGN KEY (SID) REFERENCES Serie (SID),
   CONSTRAINT CK_unicitéEvalSeriePersonne UNIQUE (PID,SID),
   CONSTRAINT CK_note CHECK (note >=0 and note <=5)
   );
```

--------- ***TRIGGER*** ----------

-----trigger mise à jour du nombre d'évaluations d'une série et sa note moyenne ----------
-----trigger met à jour toutes les lignes avec la série demandée yc entre plusieurs PFs ----------
-----NbNotes = OLD.NbNotes + 1 ----------
-----le trigger ne prend pas en compte quand une évaluation est supprime (pas demandé) ----

```
create or replace  Function  FonctionNotationSerie() returns trigger as
'DECLARE
  NbNotesSerie Serie.NbNotes %TYPE;
  NoteMoySerie Serie.noteMoyenne %TYPE;
  BEGIN
     SELECT INTO NbNotesSerie,NoteMoySerie NbNotes,noteMoyenne FROM Serie WHERE (SID=NEW.SID);
     IF NbNotesSerie=0 THEN
        UPDATE Serie SET NbNotes=1,NoteMoyenne=NEW.note WHERE SID=NEW.SID;
     ELSE
        UPDATE Serie SET NbNotes = NbNotesSerie + 1  WHERE SID=NEW.SID;
        UPDATE Serie SET NoteMoyenne =(((NoteMoySerie*NbNotesSerie)+NEW.note)/NbNotes) WHERE
SID=NEW.SID;
     END IF;
  return NEW;
  END;'
LANGUAGE 'plpgsql';
```

------définition des conditions de déclenchement trigger ----------
------mise à jour du nombre de notes et de la moyenne à l'ajout et au retrait ----------
-----point non couvert : mise à jour des moyennes dans la sérue suite à suppression personne

```
CREATE TRIGGER MaJNotes_Moyennes AFTER INSERT OR UPDATE ON Evaluation
FOR EACH ROW
EXECUTE procedure FonctionNotationSerie() ;
```

---------- ***SCRIPT D'INSERTION DES NUPLETS*** ----------

```
INSERT INTO Serie(nom,plateforme) VALUES ('Squid Game','Netflix'), ('Valid','Canal
+'),('Germinal','Salto'),('Game of Thrones','OCS');

INSERT INTO Personne(nom,prenom,pseudo) VALUES
('Gamotte','Albert','AlGam'),('Zarela','Maude','mozza'),('Computing','Claude','cloud');

INSERT INTO evaluation (PID,SID,note) VALUES (1,1,4),(1,2,4),(1,3,3),(2,1,0),(2,2,3),(3,1,2);

INSERT INTO abonnes(PID,plateforme) VALUES (1,'Netflix'), (2,'Canal +');
```

-- vérifier la suppressions des relatives évaluations:
DELETE FROM personne WHERE PID = 2;

--------- ***REQUÊTES D'INTERROGATION EN SQL*** ----------

------ ÉNONCÉ A ----------
------ Quelles plateformes n'ont pas d'abonnés ? ----------
SELECT DISTINCT(plateforme)
FROM Serie
WHERE Plateforme NOT IN (SELECT plateforme FROM Abonnes);

------ ÉNONCÉ B ----------
------ Quelles personnes (en donnant son pseudo) a évalué une série ------
------ de Netflix ou une série de Canal+ ? ----------

------ requête sans l'UNION avec un OR (ou inclusif)
SELECT DISTINCT pseudo
FROM Personne as p, Evaluation as e, Serie as s
WHERE p.pid=e.pid
AND e.sid=s.sid
AND (s.plateforme='Netflix' OR s.plateforme='Canal +');

------ requête avec l'opérateur UNION
SELECT DISTINCT pseudo
FROM Personne as p, Evaluation as e, Serie as s
WHERE p.pid=e.pid
AND e.sid=s.sid
AND s.plateforme='Netflix'
UNION
SELECT DISTINCT pseudo FROM Personne as p, Evaluation as e, Serie as s
WHERE p.pid=e.pid
AND e.sid=s.sid
AND s.plateforme='Canal +';

------ >>>> *Le* TEMPS D'EXÉCUTION est plus élevé dans la requête avec l'UNION
-- On se base sur le temps prévisionnel fourni par la requête EXPLAIN ANALYSE sur PC
-- (0.222ms pour la requête sans union et 0.626ms pour la requête avec union)

------ ÉNONCÉ C ----------
------Quelles personnes (en donnant son pseudo) a évalué une série de --------
------Netflix et une série de Canal+ ? ----------

------requête sans l'intersect forme 1
SELECT DISTINCT pseudo
FROM Personne as p, Evaluation as e1,
Evaluation as e2,Serie as s1, Serie as s2
WHERE p.pid=e1.pid
AND p.pid=e2.sid
AND e1.sid=s1.sid
AND e2.sid=s2.sid
AND (s1.plateforme='Netflix' or s1.plateforme='Canal +')
AND (s2.plateforme='Netflix' or s2.plateforme='Canal +')

```
   AND s1.plateforme != s2.plateforme;
```

------requête sans l'intersect forme 2 basée sur le principe **A union B = A-(A-B)**

```
CREATE VIEW as
   SELECT DISTINCT pseudo FROM Personne as p1, Evaluation as e1
   WHERE p1.pid=e1.pid
   AND EXISTS (SELECT * FROM Serie as s1
            WHERE s1.sid = e1.sid
             AND s1.plateforme='Netflix');
CREATE VIEW B as
   SELECT DISTINCT pseudo FROM Personne as p2, Evaluation as e2
   WHERE p2.pid=e2.pid
   AND EXISTS (SELECT * FROM Serie as s2
          WHERE s2.sid = e2.sid
           AND s2.plateforme='Canal +');
--view A-B
CREATE VIEW AminusB as
   SELECT pseudo FROM A
           WHERE pseudo NOT IN (SELECT  pseudo FROM B);
--view A=A-(A-B)
 SELECT pseudo FROM A
           WHERE pseudo NOT IN (SELECT  pseudo FROM AminusB) ;
```

------ requête avec l'intersect

```
SELECT DISTINCT pseudo FROM Personne as p1, Evaluation as e1, Serie as s1
   WHERE p1.pid=e1.pid
   AND e1.sid=s1.sid
   AND s1.plateforme='Netflix'
INTERSECT
SELECT DISTINCT pseudo FROM Personne as p2, Evaluation as e2, Serie as s2
   WHERE p2.pid=e2.pid
   AND e2.sid=s2.sid
   AND s2.plateforme='Canal +';
```

------ *>>>> Le* TEMPS D'EXÉCUTION est plus élevé dans la requête sans l'INTERSECT
-- On se base sur le temps prévisionnel fourni par la requête EXPLAIN ANALYSE sur PC (1.328ms pour la requête sans intersect et 0.787ms pour la requête avec intersect)

------ ÉNONCÉ D ----------
------ Quelles séries (en donnant leur nom) ont été évaluées par au moins 2 personnes ? ------

--------1iere forme sans group by et sans exists

```
SELECT DISTINCT s.nom
FROM Serie as s, Evaluation as e1, Evaluation as e2
WHERE s.sid=e1.sid
AND s.sid=e2.sid
AND e1.pid != e2.pid;
```

--------2ieme forme sans group by mais avec exits

```
SELECT DISTINCT s.nom
FROM Serie as s, Evaluation as e1
WHERE s.sid=e1.sid
AND EXISTS (SELECT * FROM Evaluation as e2
       WHERE e2.sid=s.sid
       AND e2.pid != e1.pid);
```

-------- 3ieme forme avec group by mais sans exists
-------- hypothèse centrale : une personne ne peut noter qu'une seule fois chaque série

```
SELECT s.nom FROM Serie as s, Evaluation as e
WHERE s.sid=e.sid
GROUP BY  s.nom
HAVING COUNT (e.EID) >=2;
```

------ >>>> *Le* TEMPS D'EXÉCUTION est le plus élevé dans la requête 'sans group by et sans exists',
-- le moins élevé dans la requête 'avec GROUP BY et sans EXITS'
-- On se base sur le temps prévisionnel fourni par la requête EXPLAIN ANALYSE sur PC
-- (0.344ms pour la requête sans group by et sans exists,
-- 0.279ms pour la requête sans group by mais avec exits,
-- 0.219ms pour la requête avec group by mais sans exists)

------ *ÉNONCÉ E* ----------
------ Quelles séries (en donnant leur nom) ont été évaluées par toutes les personnes ----------
------ de la base de données ? ----------

------ requête sans GROUP BY 1ier version

```
SELECT DISTINCT nom
  FROM serie
  WHERE nom NOT IN (
                    SELECT s.nom
                    FROM serie s
                    CROSS JOIN Personne p
                    LEFT JOIN evaluation e
                    ON (e.PID=p.PID AND e.SID=s.SID)
                    WHERE e.EID IS NULL);
```

------ requête sans GROUP BY 2ieme version avec la division

```
SELECT s.nom FROM Serie as s
WHERE NOT EXISTS (SELECT * FROM Personne as p
                            WHERE NOT EXISTS (SELECT * FROM Evaluation as e
                                                    WHERE e.pid=p.pid
                                                    AND  e.sid=s.sid));
```

------ requête avec GROUP BY

```
SELECT s.nom
  FROM evaluation e
  LEFT JOIN serie s ON s.SID = e.SID
  GROUP BY s.nom
  HAVING COUNT (DISTINCT e.PID)=
                                (SELECT COUNT(DISTINCT PID)
                                        FROM personne);
```

------ >>>> *Le* TEMPS D'EXÉCUTION est le plus élevé dans la requête 'sans GROUP BY'
-- On se base sur le temps prévisionnel fourni par la requête EXPLAIN ANALYSE sur PC
-- (0.293ms pour la requête 'sans GROUP BY',
-- 0.227ms pour la requête 'avec GROUP BY')

------ *ÉNONCÉ F* ----------
------ Quelles séries (en donnant leur nom) sont les moins bien notées ? ----------

------ 1iere interprétation de la requête : les séries dont les notes sont les plus basses

```
SELECT s.nom FROM Serie as s, Evaluation as e1
```

```
WHERE s.sid=e1.sid
AND e1.note is not null
AND e1.note <= ALL (SELECT note FROM  Evaluation WHERE note IS NOT NULL);
```

------ 2ieme interprétation de la requête : les séries dont les notes moyennes sont les plus basses

```
SELECT nom
 FROM serie
 WHERE NbNotes > 0
 AND noteMoyenne = (SELECT MIN(noteMoyenne)
                    FROM serie
                     WHERE NbNotes > 0);
```

------ *ÉNONCÉ G* ----------

------ Quel est le nombre de notes par série en ne tenant compte que ----------

------ des notes données par les abonnés de la plateforme diffusant la série ? ----------

```
SELECT s.plateforme, s.nom, COUNT(*) AS nbnotesabonnes
 FROM abonnes a
 INNER JOIN evaluation e ON e.pid = a.pid
 INNER JOIN serie s ON s.sid = e.sid
 WHERE a.plateforme = s.plateforme
 GROUP BY s.plateforme, s.nom;
```

------ *ÉNONCÉ H* ----------

------ Quel est le nombre de notes par série en séparant les notes données ----------

------ par les abonnés de la plateforme diffusant la série et les notes des personnes -----

------ non abonnées à la plateforme ? ----------

------ 1iere approche prenant comme hypothèse qu'une même personne n'a qu'un seul abonnement à une ---plate forme----------

------Principe du compte des notes des abonnés d'une série : seules les notes des abonnés de cette série y sont —---comptabilisées

```
SELECT
 s.plateforme,
 s.nom,
 SUM(CASE
  WHEN (a.plateforme = s.plateforme AND eid IS NOT NULL) THEN 1
  ELSE 0
  END) as nbnotesabonnes,
 SUM(CASE
  WHEN  eid IS NOT NULL AND a.plateforme IS NULL
                THEN 1
   ELSE 0
   END) AS nbnotesnonabonnes
 FROM serie s
 LEFT JOIN evaluation e ON e.sid = s.sid
 LEFT JOIN abonnes a ON (a.pid = e.pid AND a.plateforme = s.plateforme)
 GROUP BY s.plateforme, s.nom;
```

----2ieme approche pour tenir compte du cas où une personne qui a évalué une série d'une plateforme

----peut avoir un 2ieme abonnement à cette plateforme ----------

---1ier etape création d'une table avec les évaluations des seuls abonnés des séries concernées

Create view WWW as
SELECT s.sid,s.nom,s.plateforme,e.eid FROM serie s, evaluation e, abonnes a
WHERE e.sid = s.sid
AND a.pid=e.pid
AND s.plateforme=a.plateforme
GROUP BY s.sid,s.plateforme,e.eid;

------> 2ieme étape : rajout des toutes les évaluations yc. celles de personnes non abonnées aux (Serie,PF) – – - -concernées
----------et on fait le décompte sur la base des eid pour distinguer les cas des abonnés des autres

SELECT s.nom,s.plateforme,
SUM(CASE WHEN w.eid IS NOT NULL THEN 1
ELSE 0
END) as nbnotesabonnes,
SUM (CASE WHEN w.eid IS NULL THEN 1
ELSE 0
END) as nbnotesnonabonnes
FROM Evaluation e1
JOIN Serie s on e1.sid=s.sid
LEFT OUTER JOIN WWW w ON w.eid=e1.eid
GROUP BY s.nom,s.plateforme;

--------- ***REQUÊTES EN ALGÈBRE RELATIONELLE*** ----------

------ *ÉNONCÉ A* ----------
→ $\pi_{plateforme}$ (Serie)- $\pi_{plateforme}$ (Abonnes)

------ *ÉNONCÉ B* ----------

------ *version sans l'union*

→ π_{pseudo} ($\sigma_{plateforme='Netflix' \text{ \& } 'Canal +'}$ ((Serie $_{sid->s.sid}$) |X| $_{s.sid=e.sid}$ ($\pi_{pseudo,e.sid}$ ((Personne $_{pid->p.pid}$) |X| $_{p.pid=e.pid}$ (Evaluation $_{pid>e.pid}$))

------ *version avec l'union*

→ $A = \pi_{pseudo}$ ($\sigma_{PF='Netflix'}$ ((Serie $_{sid->s.sid}$) |X| $_{s.sid=e.sid}$ ($\pi_{pseudo,e.sid}$ ((Personne $_{pid->p.pid}$) |X|$_{a.pid=e.pid}$(Evaluation $_{pid->e.pid}$))

→ $B = \pi_{pseudo}$ ($\sigma_{PF='Canal +'}$ (Serie $s_{sid->s.sid}$) |X| $_{s.sid=e.sid}$ ($\pi_{pseudo,e.sid}$ ((Personne $_{pid->p.pid}$) |X|$_{a.pid=e.pid}$(Evaluation $_{pid->e.pid}$))

→ $A \cup B$

------ *ÉNONCÉ C* ----------

→ $A = \pi_{pseudo}$ ($\sigma_{PF='Netflix'}$ ((Serie $_{sid->s.sid}$) |X|$_{s.sid=e.sid}$ ($\pi_{pseudo,e.sid}$ ((Personne $_{pid->p.pid}$) |X|$_{a.pid=e.pid}$(Evaluation $_{pid>e.pid}$))

→ $B = \pi_{pseudo}\ (\sigma_{PF='Canal+'}\ (Serie\ s_{sid->s.sid})\ |X|_{s.sid=e.sid}\ (\pi_{pseudo,e.sid}\ ((Personne_{pid->p.pid})\ |X|_{a.pid=e.pid} (Evaluation_{pid>e.pid}))$

→ **A ∩ B**

------ *ÉNONCÉ D* ----------

→ $C = (Evaluation_{sid->e2.sid})\ |X|_{e2.sid=e.1sid}\ (Evaluation_{sid->e1.sid})$

→ $D = (\pi_{e1.sid}\ (\sigma_{e2.pid\ !=e1.pdd}\ (C)))$

→ $E = \pi_{pseudo}\ (Serie_{sid->s.sid})\ |X|_{s.sid=e1.sid} (D)$

------ *ÉNONCÉ E* ----------

→ $\pi_{nom}\ (Serie) - \pi_{nom}\ (\sigma_{eid\ is\ null}(\ (Serie_{sid->s.sid})\ X\ (Personne_{pid->p.pid})\ )\ ]\ |X|_{p.pid=e.pid\ and\ s.sid=e.sid} (Evaluation_{sid->e.sid}))$