كلية العلوم
FACULTÉ DES SCIENCES

Filière :

Analytique des données et intelligence artificielle

COMPTE RENDU TP 3:

Base de données

Réalisé par :
OUJAID Soumia

Professeur :
Mr OUKBACH Yassine

Année universitaire : 2023/2024

Objectifs :

Le travail pratique 3 du module de Bases de données et modélisation se concentrent sur l'utilisation des requêtes avancées en SQL. Les objectifs principaux sont :

- ⇨ l'utilisation de requêtes multi-tables.
- ⇨ les opérateurs de jointures.
- ⇨ les opérateurs ensemblistes.
- ⇨ les requêtes imbriquées.
- ⇨ les concepts de groupement et agrégation.

Exécution des tâches :

1. Pour la 1 ère question il nous a demandé d'ajouter un champ « date_naissance » à la table étudiant et y insérer des valeurs en utilisant le navigateur ou des opérations update. Mais dans cette table « étudiant » qu'on a créé, ce champ existe déjà.

2. **Affichage du nom, prénom et date de naissance des étudiants :**
   Une requête « Select » a été utilisée pour afficher les informations de l'étudiant de tous les étudiants de la table.

```
SQL> Select NomEtudiant , PrenomEtudiant , DateNaissance
  2  From Etudiant ;

NOMETUDIANT          PRENOMETUDIANT       DATENAIS
-------------------- -------------------- --------
Oujaid               Soumia               21/10/02
El Maati             Karima               13/09/02
El Nabolssi          Foad                 14/09/03
Kassmi               Salim                02/11/04
El Maati             Souaad               23/02/99

SQL> _
```

3. **Calcul de l'âge moyen des étudiants :**
   L'âge moyen des étudiants a été calculé de deux manières différentes.
   Extract(year From Current_date) est utilisé a fin d'extraire l'année actuelle.
   Extract (year From DateNaissance) est utilisé pour retourner l'année de naissance de l'étudiant.

```
SQL> Select avg( extract (year From Current_date) - extract (year From DateNaissance)) as Age_Moyen
  2  From Etudiant;

 AGE_MOYEN
----------
        22

SQL> Select avg( Months_between( Sysdate , DateNaissance) /12) as Age_Moyen
  2  From Etudiant;

 AGE_MOYEN
----------
21,3708564

SQL> _
```

4. **Affichage du nombre d'étudiants et de l'âge moyen :**

Le nombre d'étudiants de l'établissement et leur âge moyen ont été affichés.
Count (NomEtudiant) est une fonctionne utilisée pour calculer le nombre des étudiants existent dans la table à l'aide de l'attribut NomEtudiant.
Avg (année) est utilisée pour calculer le moyen d'âge de tous les étudiants insérés.

```
SQL> Select count( NomEtudiant ) as Nb_Etud,
  2  Avg( extract (year From Current_date) - extract( year From DateNaissance)) as Age_Moyen
  3  From Etudiant;

   NB_ETUD  AGE_MOYEN
---------- ----------
         5         22

SQL>
```

5. **Affichage du nombre, minimum, maximum et moyenne d'âge des étudiants :**

Les statistiques d'âge, incluant le nombre d'étudiants, le minimum, le maximum et la moyenne, ont été affichées.
Max (année) retourne l'âge d'étudiant le plus grand de la table.
Min (année) retourne l'âge d'étudiant le plus petit de la table.

```
SQL> Select count( NomEtudiant ) as Nb_Etud ,
  2  Avg( extract (year From Current_date) - extract( year From DateNaissance)) as Age_Moyen ,
  3  Max( extract (year From Current_date) - extract( year From DateNaissance)) as Age_Max ,
  4  Min( extract (year From Current_date) - extract( year From DateNaissance)) as Age_Min
  5  From Etudiant ;

   NB_ETUD  AGE_MOYEN    AGE_MAX    AGE_MIN
---------- ---------- ---------- ----------
         5         22         25         20

SQL>
```

6. **Liste par année le nombre d'étudiants et les statistiques des notes :**

Les statistiques des notes, y compris le nombre d'étudiants, la moyenne, le maximum et le minimum, ont été listées par année.
Ici on a fait appel à la notion de jointure a fin d'assembler les données dont on aurait besoin des deux tables Résultat et Etudiant.
La clause Group by (année) est utilisée pour regrouper les résultats en fonctionne de l'année de naissance des étudiants.
La clause Order By (année) est utilisée pour trier les résultats en fonctionne de l'année de naissance des étudiants.

```
SQL> Select extract( year From e.DateNAissance) as Annee ,
  2  count( distinct e.CodeEtudiant) as Nb_Etud ,
  3  Avg ( r.Note ) as Moyen_Note ,
  4  Max ( r.Note ) as Max_Note ,
  5  Min ( r.Note ) as Min_Note
  6  From Etudiant e Join Resultat r On e.CodeEtudiant = r.CodeEtudiant
  7  Group by extract( year From e.DateNAissance)
  8  Order by extract( year From e.DateNAissance);

    ANNEE    NB_ETUD MOYEN_NOTE   MAX_NOTE   MIN_NOTE
--------- ---------- ---------- ---------- ----------
     1999          1         15         15         15
     2002          2      12,25       14,5         10
     2003          1         11         11         11
     2004          1         19         19         19

SQL>
```

7. Liste par année du nombre d'étudiants des années 1 et 2 avec statistiques des notes :

Les statistiques des notes pour les étudiants de l'année 1 et 2 ont été affichées par année.

Dans mes 1ere insertions, j'ai ajouté les deux années 1999 et 2002. Signification pour Where (Annee) in (1999 , 2002) .

```
SQL> Select extract( year From e.DateNAissance) as Annee ,
  2  count( distinct e.CodeEtudiant) as Nb_Etud ,
  3  Avg ( r.Note ) as Moyen_Note ,
  4  Max ( r.Note ) as Max_Note ,
  5  Min ( r.Note ) as Min_Note
  6  From Etudiant e Join Resultat r On e.CodeEtudiant = r.CodeEtudiant
  7  Where ( extract( year From e.DateNAissance)) in (1999 , 2002)
  8  Group by extract( year From e.DateNAissance)
  9  Order by extract( year From e.DateNAissance);

    ANNEE    NB_ETUD MOYEN_NOTE   MAX_NOTE   MIN_NOTE
--------- ---------- ---------- ---------- ----------
     1999          1         15         15         15
     2002          2      12,25       14,5         10

SQL>
```

8. Recherche par année du nombre d'étudiants d'Agadir et Casa avec statistiques des notes :

Les statistiques des notes pour les étudiants d'Agadir et Casa ont été affichées par année.

```
SQL> Select extract( year From e.DateNAissance) as Annee ,
  2  count( distinct e.CodeEtudiant) as Nb_Etud ,
  3  Avg ( r.Note ) as Moyen_Note ,
  4  Max ( r.Note ) as Max_Note ,
  5  Min ( r.Note ) as Min_Note
  6  From Etudiant e Join Resultat r On e.CodeEtudiant = r.CodeEtudiant
  7  Where Ville in ('Casa' , 'Agadir')
  8  Group by extract( year From e.DateNAissance)
  9  Order by extract( year From e.DateNAissance);

    ANNEE    NB_ETUD MOYEN_NOTE   MAX_NOTE   MIN_NOTE
--------- ---------- ---------- ---------- ----------
     2002          1       14,5       14,5       14,5
     2004          1         19         19         19

SQL> _
```

9. Liste par année du nombre d'étudiants des années 1 et 2 de Marrakech ou Taroudant avec statistiques des notes :

Les statistiques des notes pour les étudiants de l'année 1 et 2 de Marrakech ou Taroudant ont été listées par année.

```
SQL> Select extract( year From e.DateNAissance) as Annee ,
  2  count( distinct e.CodeEtudiant) as Nb_Etud ,
  3  Avg ( r.Note ) as Moyen_Note ,
  4  Max ( r.Note ) as Max_Note ,
  5  Min ( r.Note ) as Min_Note
  6  From Etudiant e Join Resultat r On e.CodeEtudiant = r.CodeEtudiant
  7  Where ( extract( year From e.DateNAissance)) in (1999 , 2002)
  8  And Ville in ('Casa' , 'Agadir')
  9  Group by extract( year From e.DateNAissance)
 10  Order by extract( year From e.DateNAissance);

    ANNEE    NB_ETUD MOYEN_NOTE   MAX_NOTE   MIN_NOTE
--------- ---------- ---------- ---------- ----------
     2002          1       14,5       14,5       14,5

SQL>
```

10. Liste par année du nombre d'étudiants avec moyenne entre 12 et 16 :

Les étudiants avec une moyenne comprise entre 12 et 16 ont été listés par année.

La clause Having est utilisée pour filtrer les résultats agrégés basés sur une condition spécifiée. Comme Where juste dans Having on peut utiliser les fonctionnes (Avg, count, min…) au contraire de Where.

```
SQL> Select extract( year From e.DateNAissance) as Annee ,
  2  count( distinct e.CodeEtudiant) as Nb_Etud ,
  3  Avg ( r.Note ) as Moyen_Note ,
  4  Max ( r.Note ) as Max_Note ,
  5  Min ( r.Note ) as Min_Note
  6  From Etudiant e Join Resultat r On e.CodeEtudiant = r.CodeEtudiant
  7  Group by extract( year From e.DateNAissance)
  8  Having Avg( r.Note ) between 12 And 16
  9  Order by extract( year From e.DateNAissance);

    ANNEE    NB_ETUD MOYEN_NOTE   MAX_NOTE   MIN_NOTE
--------- ---------- ---------- ---------- ----------
     1999          1         15         15         15
     2002          2      12,25       14,5         10

SQL> _
```

11.Lister par année le nombre d'étudiants, la moyenne, le maximum et le minimum de notes ordonnés par moyenne des notes :

Liste les statistiques des notes ordonnées par moyenne des notes par année.

```
SQL> Select extract( year From e.DateNAissance) as Annee ,
  2  count( distinct e.CodeEtudiant) as Nb_Etud ,
  3  Avg ( r.Note ) as Moyen_Note ,
  4  Max ( r.Note ) as Max_Note ,
  5  Min ( r.Note ) as Min_Note
  6  From Etudiant e Join Resultat r On e.CodeEtudiant = r.CodeEtudiant
  7  Group by extract( year From e.DateNAissance)
  8  Order by Moyen_Note ;

    ANNEE    NB_ETUD MOYEN_NOTE   MAX_NOTE   MIN_NOTE
--------- ---------- ---------- ---------- ----------
     2003          1         11         11         11
     2002          2      12,25       14,5         10
     1999          1         15         15         15
     2004          1         19         19         19

SQL>
```

12.Afficher par année le nombre d'étudiants, la moyenne, le maximum et le minimum de notes ordonnés par nombre d'étudiants et par moyenne de notes descendantes :

Affiche les statistiques des notes ordonnées d'abord par nombre d'étudiants, puis par moyenne des notes de manière descendante par année.

La ligne « Order By Nb_Etud, Moyen_Note desc ; » indique que les résultats doivent être triés en fonction du nombre d'étudiants de manière ascendante, et si deux résultats ont le même nombre d'étudiants, ils doivent être triés en fonction de la moyenne des notes de manière descendante.

```
SQL> Select extract( year From e.DateNAissance) as Annee ,
  2  count( distinct e.CodeEtudiant) as Nb_Etud ,
  3  Avg ( r.Note ) as Moyen_Note ,
  4  Max ( r.Note ) as Max_Note ,
  5  Min ( r.Note ) as Min_Note
  6  From Etudiant e Join Resultat r On e.CodeEtudiant = r.CodeEtudiant
  7  Group by extract( year From e.DateNAissance)
  8  Order by Nb_Etud , Moyen_Note desc ;

    ANNEE    NB_ETUD MOYEN_NOTE   MAX_NOTE   MIN_NOTE
--------- ---------- ---------- ---------- ----------
     2004          1         19         19         19
     1999          1         15         15         15
     2003          1         11         11         11
     2002          2      12,25       14,5         10

SQL> _
```

13.Donner la liste de noms des enseignants dont le nom contient ‘i’ à la fin :

Donne la liste des enseignants dont le nom se termine par 'i' avec la première lettre en majuscule et les autres en minuscules.

La fonction InitCap(NomEnseignant) pour formater la première lettre de chaque mot dans la colonne NomEnseignant de la table Enseignant, ceci en mettant le première lettre du nom en majuscule et le reste en minuscule.
La syntaxe %i signifie que la chaîne de caractères doit se terminer par ‘i’, et le caractère % représente n'importe quelle séquence de caractères avant ‘i’.

```
SQL> Select InitCap( NomEnseignant )
  2  From Enseignant
  3  Where NomEnseignant Like '%i';

INITCAP(NOMENSEIGNAN
--------------------
Jamoli
Oufedi
Solami
Houari

SQL>
```

14.Donner le nom des enseignants qui assurent plus de deux cours :

Donne le nom des enseignants qui assurent plus de deux cours. Ma requête renvoie une liste vide car dans mon table des enseignants chaque prof assure juste à un seul cours.

```
SQL> Select NomEnseignant
  2  From Enseignant e Join Charge c
  3  On e.CodeEnseignant = c.CodeEnseignant
  4  Group By NomEnseignant
  5  Having count( c.CodeCours ) >2;

aucune ligne sÚlectionnÚe

SQL>
```

15.Afficher le nom et la spécialité des enseignants dont la spécialité est connue :

Affiche le nom et la spécialité des enseignants dont la spécialité est renseignée.
La clause Where filtre les résultats en ne renvoyant que les lignes où la colonne Spécialité n'est pas nulle. L'expression « is not null » est utilisée pour vérifier si la colonne Spécialité a une valeur non nulle.

```
SQL> Select NomEnseignant
  2  From Enseignant
  3  Where Specialite is not Null;

NOMENSEIGNANT
--------------------
Jamoli
Oufedi
Boulouz
Solami
Houari
Mouradi

6 lignes sÚlectionnÚes.

SQL> _
```

16.Lister les noms des enseignants ayant la même spécialité :

Liste les enseignants qui ont la même spécialité.

La clause Group By Spécialité regroupe les enregistrements par spécialité, et la clause Having count(*) > 1 filtre les groupes pour ne renvoyer que ceux avec plus d'un enregistrement.

```
SQL> Select NomEnseignant
  2  From Enseignant
  3  Where Specialite In (
  4  Select Specialite
  5  From Enseignant
  6  Group By Specialite
  7  Having count( * ) >1 )
  8  Group By NomEnseignant;

NOMENSEIGNANT
--------------------
Jamoli
Mouradi

SQL> _
```

17.Lister les noms des enseignants qui font le même cours :

Liste les enseignants qui enseignent le même cours.

« Join Charge ch1 on e.CodeEnseignant = ch1.CodeEnseignant » ici on a une jointure relie la table Enseignant à la table Charge (ch1) en utilisant la colonne CodeEnseignant.

« Join Charge ch2 ON ch1.CodeCours = ch2.CodeCours » représente aussi une jointure qu'est effectuée à la table Charge (ch1) avec la table Charge (ch2).

« ch1.CodeEnseignant <> ch2.CodeEnseignant » garantit que l'enseignant dans la première charge (ch1) est différent de celui dans la deuxième charge (ch2)

```
SQL> Select NomEnseignant
  2  From Enseignant e
  3  Join Charge ch1 on e.CodeEnseignant = ch1.CodeEnseignant
  4  Join Charge ch2 on ch1.CodeCours = ch2.CodeCours
  5  And ch1.CodeEnseignant <> ch2.CodeEnseignant
  6  Join Cours c on ch1.CodeCours = c.CodeCours
  7  Group By NomEnseignant;

aucune ligne sÚlectionnÚe
```

Autre Méthode :

```
SQL> Select E1.NomEnseignant , E2.NomEnseignant
  2  From Enseignant E1 , Enseignant E2
  3  Where E1.CodeEnseignant < E2.CodeEnseignant And (E1.CodeEnseignant , E2.CodeEnseignant ) In (
  4     Select C1.CodeEnseignant , C2.CodeEnseignant
  5     From Charge C1 , Charge C2
  6     Where C1.CodeCours = C2.CodeCours
  7  )
  8  Group By E1.NomEnseignant , E2.NomEnseignant ;

aucune ligne sÚlectionnÚe

SQL>
```

18. Créer une autre table ETUDIANTS semblable à Etudiant et la charger par des tuples dont les num_etu sont distinct de ceux d'ETUDIANT. Faire l'union d'ETUDIANTS et ETUDIANT :

Crée une nouvelle table ETUDIANTS similaire à la table Etudiant et effectue l'union des deux tables.

1ere Méthode pour créer la table ETUDIAN :

```
SQL> create table ETUDIANT1(
  2     CodeEtudiant varchar2(10) ,
  3     NomEtudiant varchar2(20) ,
  4     PrenomEtudiant varchar2(20),
  5     DateNaissance date ,
  6     Ville varchar2(20)
  7  );

Table crÚÚe.

SQL>
```

2ème Méthode de création de table ETUDIANT pour quel porte les mêmes champs que Etudiant :

« Where 1=0 » est fait pour que l'ETUDIANTS ne porte pas les données insérer dans la table Etudiant.

```
SQL> Create table ETUDIANTS as
  2     Select *
  3     From Etudiant
  4     Where 1 = 0;

Table crÚÚe.

SQL>
```

Les insertions nécessaires :

```
SQL> insert into ETUDIANTS (CodeEtudiant , NomEtudiant , PrenomEtudiant , DateNaissance , Ville)
  2  values ('Et1' , 'Doma', 'Adam' , '2-8-2001' , 'Saouira');

1 ligne crÚÚe.

SQL> insert into ETUDIANTS (CodeEtudiant , NomEtudiant , PrenomEtudiant , DateNaissance , Ville)
  2  values ('Et2' , 'Fadoua', 'Samira' , '31-10-2005' , 'Agadir');

1 ligne crÚÚe.

SQL> insert into ETUDIANTS (CodeEtudiant , NomEtudiant , PrenomEtudiant , DateNaissance , Ville)
  2  values ('Et3' , 'Fadili', 'Fatima' , '23-11-2003' , 'Casa');

1 ligne crÚÚe.

SQL>
```

UNION des deux tables Etudiant et ETUDIANS :

```
SQL> Select * From Etudiant
  2  UNION
  3  Select * From ETUDIANTS;

CODEETUDIA NOMETUDIANT          PRENOMETUDIANT       DATENAIS VILLE
---------- -------------------- -------------------- -------- ----------
01         Oujaid               Soumia               21/10/02 Agadir
02         El Maati             Karima               13/09/02 Taroudant
03         El Nabolssi          Foad                 14/09/03 Rabat
04         Kassmi               Salim                02/11/04 Casa
05         El Maati             Souaad               23/02/99 Fes
Et1        Doma                 Adam                 02/08/01 Saouira
Et2        Fadoua               Samira               31/10/05 Agadir
Et3        Fadili               Fatima               23/11/03 Casa

8 lignes sÚlectionnÚes.

SQL>
```

19. Faire l'union d'Etudiant et d'ETUDIANTS pour l'année 1 :

Effectue l'union des étudiants des tables ETUDIANT et ETUDIANTS pour l'année 1.

Cette requête renvoie les années de naissance uniques des étudiants dont les CodeEtudiant sont respectivement '01' dans la table Etudiant et 'Et1' dans la table ETUDIANTS.

Union assure que chaque année de naissance apparaît une seule fois dans le résultat final.

```
SQL> Select extract( year From DateNaissance) as Annee
  2  From Etudiant e
  3  Where e.DateNaissance In(
  4     Select DateNaissance
  5     From Etudiant
  6     Where CodeEtudiant in ('01')
  7  )UNION
  8  Select extract ( year From DateNaissance) as Annee
  9  From ETUDIANTS e
 10  Where e.DateNaissance In(
 11     Select DateNaissance
 12     From ETUDIANTS
 13     Where CodeEtudiant in ('Et1')
 14  );

     ANNEE
----------
      2002
      2001

SQL>
```

20. Faire l'union de : (les étudiants de la 1 année de ETUDIANT) et (les étudiants de la 3 année de ETUDIANTS) :

Effectue l'union des étudiants de la 1ère année de la table ETUDIANT et des étudiants de la 3ème année de la table ETUDIANTS.

```
SQL> Select extract( year From DateNaissance) as Annee
  2  From Etudiant e
  3  Where e.DateNaissance In(
  4     Select DateNaissance
  5     From Etudiant
  6     Where CodeEtudiant in ('01')
  7  )UNION
  8  Select extract ( year From DateNaissance) as Annee
  9  From ETUDIANTS e
 10  Where e.DateNaissance In(
 11     Select DateNaissance
 12     From ETUDIANTS
 13     Where CodeEtudiant in ('Et3')
 14  );

    ANNEE
----------
      2002
      2003

SQL>
```

21. Donner le nom, la moyenne, le minimum et le maximum de notes de chaque étudiant :

Donne le nom, la moyenne, le minimum et le maximum des notes de chaque étudiant.

« Natural Join» est une forme de jointure implicite qui relie automatiquement les colonnes ayant le même nom dans les deux tables. Dans ce cas, c'est la colonne CodeEtudiant entre les deux tables Etudiant et Résultat.

```
SQL> Select NomEtudiant ,
  2     Min( Note ),
  3     Max( Note ),
  4     Avg( Note )
  5  From Etudiant natural join Resultat
  6  Group By CodeEtudiant , NomEtudiant ;

NOMETUDIANT          MIN(NOTE)  MAX(NOTE)  AVG(NOTE)
-------------------- ---------- ---------- ----------
Oujaid                     14,5       14,5       14,5
El Maati                     10         10         10
El Nabolssi                  11         11         11
Kassmi                       19         19         19
El Maati                     15         15         15

SQL>
```

22. Donner le nom la moyenne le minimum et le maximum de notes de chaque étudiant de la 1ère année :

Donne le nom, la moyenne, le minimum et le maximum des notes de chaque étudiant de 1ère année.

```
SQL> Select NomEtudiant as Nom,
  2     Min (Note) as Min,
  3     Max (Note) as Max,
  4     Avg (Note) as Moyen
  5  From Etudiant natural join Resultat
  6  Where (extract (year From DateNaissance)) in ('2002')
  7  Group By CodeEtudiant , NomEtudiant ;

NOM                        MIN        MAX      MOYEN
-------------------- ---------- ---------- ----------
Oujaid                    14,5       14,5       14,5
El Maati                    10         10         10

SQL> _
```

23. Donner le nom la moyenne le minimum et le maximum de notes de chaque étudiant dont la moyenne est supérieure à 11.

Donne le nom, la moyenne, le minimum et le maximum des notes de chaque étudiant dont la moyenne >11.

```
SQL> Select NomEtudiant as Nom,
  2     Min (Note) as Min,
  3     Max (Note) as Max,
  4     Avg (Note) as Moyen
  5  From Etudiant natural join Resultat
  6  Group By CodeEtudiant , NomEtudiant
  7  Having Avg( Note ) > 11;

NOM                          MIN        MAX      MOYEN
-------------------- ---------- ---------- ----------
Oujaid                     14,5       14,5       14,5
Kassmi                       19         19         19
El Maati                     15         15         15

SQL> _
```

24. Donner le nom la moyenne le minimum et le maximum de notes de chaque étudiant de la 1 année et dont la moyenne est supérieure 12.

Donne le nom, la moyenne, le minimum et le maximum des notes de chaque étudiant de 1ère année et de moyenne >12.

```
SQL> Select NomEtudiant as Nom,
  2     Min (Note) as Min,
  3     Max (Note) as Max,
  4     Avg (Note) as Moyen
  5  From Etudiant natural join Resultat
  6  Where extract(year From DateNaissance) In (
  7     Select  extract(year From DateNaissance)
  8     From Etudiant
  9     Where CodeEtudiant in ('01')
 10  )
 11  Group  By CodeEtudiant , NomEtudiant
 12  Having Avg( Note ) > 12;

NOM                          MIN        MAX      MOYEN
-------------------- ---------- ---------- ----------
Oujaid                     14,5       14,5       14,5

SQL> _
```

25. Lister le numéro, le nom et la moyenne de chaque étudiant.

Regroupe le numéro (table Etudiant) et la moyenne (table Résultat) de chaque étudiant.

```
SQL> Select CodeEtudiant as Num ,
  2     NomEtudiant as Nom ,
  3     Avg( Note ) as Moyen
  4  From Etudiant natural join Resultat
  5  Group By CodeEtudiant , NomEtudiant ;

NUM        NOM                       MOYEN
---------- -------------------- ----------
01         Oujaid                     14,5
02         El Maati                     10
03         El Nabolssi                  11
04         Kassmi                       19
05         El Maati                     15

SQL> _
```

26. Afficher les enseignants qui assurent tous les cours sauf le cours de réseaux.

Sélectionne les informations des enseignants dont la spécialité n'est pas réseaux.

```
SQL> Select * From Enseignant
  2  Where Specialite <> 'Réseaux';

CODEENSEIG NOMENSEIGNANT        PRENOMENSEIGNANT     SPECIALITE
---------- -------------------- -------------------- --------------------
E1         Jamoli               Rachid               JAVA
E2         Oufedi               Hassan               Android
E4         Solami               Loubna               Français
E5         Houari               Salma                BD
E6         Mouradi              Mehdi                JAVA

SQL>
```

27. Vérifier que les enseignants ayant une charge, figurent bien dans la table enseignant.

Récupère toutes les colonnes de la table Enseignant où il existe au moins une correspondance dans la table Charge pour le même CodeEnseignant. Voici une explication détaillée de la requête.

```
SQL> Select distinct e1.*
  2  From Enseignant e1
  3  Join Charge c on e1.CodeEnseignant = c.CodeEnseignant
  4  Where exists(
  5     Select 1
  6     From Enseignant e2
  7     Where e2.CodeEnseignant = e1.CodeEnseignant
  8  );

CODEENSEIG NOMENSEIGNANT        PRENOMENSEIGNANT     SPECIALITE
---------- -------------------- -------------------- --------------------
E1         Jamoli               Rachid               JAVA
E2         Oufedi               Hassan               Android
E3         Boulouz              Brahim               Réseaux
E4         Solami               Loubna               Français
E5         Houari               Salma                BD

SQL>
```

Autre Méthode :

Récupère les codes d'enseignant (CodeEnseignant) à partir de la table Charge où il n'existe aucune correspondance dans la table Enseignant pour le même CodeEnseignant.

```
SQL> Select CodeEnseignant
  2  From Charge c
  3  Where not exists(
  4     Select CodeEnseignant
  5     From Enseignant
  6     Where CodeEnseignant = c.CodeEnseignant
  7  );

aucune ligne sÚlectionnÚe

SQL>
```

28.Construire une vue qui contient les numéros, les noms et les moyennes des étudiants de l'année 1 :

(VueEtudiant) contient les colonnes num, nom, et moyen, qui représentent respectivement le CodeEtudiant, le NomEtudiant et Avg(Note) comme moyenne des notes pour les étudiants ayant la même date de naissance que l'étudiant avec le code '01'.

```
SQL> Create View VueEtudiant as
  2     Select e.CodeEtudiant as num,
  3            e.NomEtudiant as nom,
  4            Avg(r.Note) as moyen
  5     From Etudiant e Join Resultat r on e.CodeEtudiant = r.CodeEtudiant
  6     Where e.DateNaissance in (select DateNaissance From Etudiant Where CodeEtudiant ='01')
  7     Group by e.CodeEtudiant , e.NomEtudiant ;

Vue crÚÚe.

SQL>
```

29.Afficher le contenu de la vue créée à la question précédente :

Donne le contenu de VueEtudiant créée.

```
SQL> Select *
  2  From VueEtudiant;

NUM        NOM                       MOYEN
---------- -------------------- ----------
01         Oujaid                     14,5

SQL> _
```

Synthèse :

Ce TP en langage SQL à couvert plusieurs aspects avancés de la manipulation des données, notamment l'utilisation de requêtes multi-tables, les opérations de jointures, les opérations ensemblistes, les requêtes imbriquées, ainsi que le groupement et l'agrégation des données.

En général, ce travail a permis de mettre en pratique les connaissances en SQL, en résolvant des problèmes de plus en plus complexes. L'utilisation de différentes clauses SQL a permis de traiter des requêtes variées, couvrant des aspects tels que le calcul de statistiques, la vérification de l'intégrité des données, et la création de vues pour simplifier l'accès à l'information. Cela renforce la compréhension des étudiants sur la manière d'interagir avec une base de données relationnelle en utilisant le langage SQL.