TRAVAILLE A RENDRE

TP3: BASE DE DONNEES **NOM ET PRENOM :IZIKKI Hajar**

PROF ENCADRANT : M. OUKDACH

1. Ajouter le champ date_naissance à la table étudiant et y insérer des valeurs en utilisant le navigateur ou des opérations update :

```
SQL> UPDATE ETUDIANT SET datenaissance = TO_DATE('1990-05-15', 'YYYY-MM-DD') WHERE nomEtudiant='Dupont' ;

1 ligne mise Ó jour.

SQL>
```

2. Afficher le nom, le prénom et la date de naissance (le nom du jour, celui du mois et l'année) des étudiants.

```
SQL> SELECT nomEtudiant,prenomEtudiant, TO_CHAR(dateNaissance, 'Day, Month DD, YYYY') AS date_de_naissance FROM ETUDIANT;

NOMETUDIANT          PRENOMETUDIANT
-------------------- --------------------
DATE_DE_NAISSANCE
--------------------------------------------------------------------------------
Dupont               Jean
Mardi    , Mai       15, 1990

Martin               Sophie
Vendredi, Ao¹t       21, 1992

Dubois               Pierre
Dimanche, Mars       10, 1991


NOMETUDIANT          PRENOMETUDIANT
-------------------- --------------------
DATE_DE_NAISSANCE
--------------------------------------------------------------------------------
Lefevre              Marie
Vendredi, Novembre   05, 1993

Moreau               François
Jeudi    , Juillet   28, 1994


SQL>
```

TO_CHAR(date_naissance, 'Day, DD Month YYYY') : utilise la fonction TO_CHAR pour formater la date de naissance. Elle affiche le nom du jour (Day), le jour du mois (DD), le nom du mois (Month) et l'année (YYYY).

3. Calculer l'âge moyen des étudiants.

```
SQL> SELECT AVG(EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM dateNaissance)) AS age_moyen FROM ETUDIANT;

 AGE_MOYEN
----------
        32

SQL>
```

- **SELECT AVG( )** : utilise la fonction **AVG** pour calculer la moyenne.
- **EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM date_naissance)** : calcule l'âge de chaque étudiant en soustrayant l'année de sa date de naissance de l'année actuelle (**SYSDATE** représente la date actuelle).

4. Afficher le nombre d'étudiants de l'établissement et le moyen de leurs âges.

```
SQL> SELECT COUNT(*) AS nombre_etudiants, AVG(EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM dateNaissance)) AS age_moyen FROM ETUDIANT;

NOMBRE_ETUDIANTS  AGE_MOYEN
---------------- ----------
               5         32

SQL>
```

SELECT COUNT(*) AS nombre_etudiants : utilise la fonction COUNT pour compter le nombre total d'enregistrements dans la table etudiants.

5. Afficher le nombre d'étudiants de l'établissement, le minimum, le maximum et la moyenne de leurs âges.

```
               5         32

SQL> SELECT
  2      COUNT(*) AS nombre_etudiants,
  3      MIN(EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM dateNaissance)) AS age_minimum,
  4      MAX(EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM dateNaissance)) AS age_maximum,
  5      AVG(EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM dateNaissance)) AS age_moyen
  6  FROM ETUDIANT;

NOMBRE_ETUDIANTS AGE_MINIMUM AGE_MAXIMUM  AGE_MOYEN
---------------- ----------- ----------- ----------
               5          30          34         32

SQL>
```

- **MIN(EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM date_naissance))** : Calcule l'âge minimum des étudiants en utilisant la fonction **MIN**
- **MAX(EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM date_naissance))** : Calcule l'âge maximum des étudiants en utilisant la fonction **MAX**.

6. Lister par année le nombre d'étudiants, la moyenne, le maximum et le minimum des notes.

```
SQL> SELECT
  2      EXTRACT(YEAR FROM datenaissance) AS annee,
  3      COUNT(codeEtudiant) AS nombre_etudiants,
  4      AVG(NOTE) AS moyenne_notes,
  5      MAX(NOTE) AS note_maximum,
  6      MIN(NOTE) AS note_minimum
  7  FROM ETUDIANT
  8  LEFT JOIN RESULTAT USING (codeEtudiant)
  9  GROUP BY EXTRACT(YEAR FROM datenaissance);

    ANNEE NOMBRE_ETUDIANTS MOYENNE_NOTES NOTE_MAXIMUM NOTE_MINIMUM
--------- --------------- ------------- ------------ ------------
     1993               1          14,2         14,2         14,2
     1990               1          18,5         18,5         18,5
     1994               1          19,3         19,3         19,3
     1992               1          16,8         16,8         16,8
     1991               1            20           20           20

SQL>
```

- **AVG(note)** : Calcule la moyenne des notes pour chaque année.
- **MAX(note)** : Trouve la note maximale pour chaque année.
- **MIN(note)** : Trouve la note minimale pour chaque année.
- **FROM ETUDIANT** : Spécifie la table des notes à partir de laquelle les données sont extraites.
- **JOIN RESULTAT**: Utilise une jointure pour associer les notes aux étudiants en fonction de l'identifiant de l'étudiant.
- **GROUP BY**: Regroupe les résultats par année.

7. Lister par année le nombre d'étudiants de l'année 1 et 2, ainsi que la moyenne, le maximum et le minimum de leurs notes.

```
SQL> SELECT
  2         EXTRACT(YEAR FROM E.dateNaissance) AS annee,
  3          COUNT(E.codeEtudiant) AS nombre_etudiants,
  4          AVG(R.note) AS moyenne_notes,
  5          MAX(R.note) AS note_maximum,
  6          MIN(R.note) AS note_minimum
  7      FROM ETUDIANT E
  8      LEFT JOIN RESULTAT R ON E.codeEtudiant = R.codeEtudiant
  9       WHERE EXTRACT(YEAR FROM E.dateNaissance) IN ('1','2')
 10      GROUP BY EXTRACT(YEAR FROM E.dateNaissance);

aucune ligne sÚlectionnÚe

SQL>
```

On va ajouter la condition à la requête précédente :

WHERE EXTRACT(YEAR FROM E.dateNaissance) IN ('1', '2'): Filtre les étudiants pour inclure uniquement ceux des années 1 et 2.

8. Chercher par année le nombre d'étudiants de la ville Agadir et Casa, ainsi que la moyenne, le maximum et le minimum de leurs notes.

```
SQL> SELECT
  2         EXTRACT(YEAR FROM E.dateNaissance) AS annee,
  3          COUNT(E.codeEtudiant) AS nombre_etudiants,
  4          AVG(R.note) AS moyenne_notes,
  5          MAX(R.note) AS note_maximum,
  6          MIN(R.note) AS note_minimum
  7      FROM ETUDIANT E
  8      LEFT JOIN RESULTAT R ON E.codeEtudiant = R.codeEtudiant
  9       WHERE E.ville IN ('Agadir','Casa')
 10      GROUP BY EXTRACT(YEAR FROM E.dateNaissance);

aucune ligne sÚlectionnÚe

SQL>
```

On a une autre condition :

WHERE e.ville IN ('Agadir', 'Casa') : Filtre les étudiants pour inclure uniquement ceux villes Agadir et Casa.

9. Lister par année le nombre d'étudiants de l'année 1 et 2, et venant de Marrakech ou de Taroudant, ainsi que la moyenne, le maximum et le minimum de leurs notes.

```
SQL> SELECT
  2         EXTRACT(YEAR FROM E.dateNaissance) AS annee,
  3          COUNT(E.codeEtudiant) AS nombre_etudiants,
  4          AVG(R.note) AS moyenne_notes,
  5          MAX(R.note) AS note_maximum,
  6          MIN(R.note) AS note_minimum
  7      FROM ETUDIANT E
  8      LEFT JOIN RESULTAT R ON E.codeEtudiant = R.codeEtudiant
  9       WHERE EXTRACT(YEAR FROM E.dateNaissance)IN ('1','2') AND E.ville IN ('Marrakech','Taroudant')
 10      GROUP BY EXTRACT(YEAR FROM E.dateNaissance);

aucune ligne sÚlectionnÚe

SQL>
```

WHERE EXTRACT(YEAR FROM E.dateNaissance) IN ('1', '2') AND e.ville IN ('Marrakech', 'Taroudant') : Filtre les étudiants pour inclure uniquement ceux des années 1 et 2, venant de Marrakech ou de Taroudant.

10. Lister par année le nombre d'étudiants, la moyenne, le maximum et le minimum de notes tels que la moyenne soit comprise entre 12 et 16.

```
SQL> SELECT
  2         EXTRACT(YEAR FROM E.dateNaissance) AS annee,
  3          COUNT(E.codeEtudiant) AS nombre_etudiants,
  4          AVG(R.note) AS moyenne_notes,
  5          MAX(R.note) AS note_maximum,
  6          MIN(R.note) AS note_minimum
  7      FROM ETUDIANT E
  8      LEFT JOIN RESULTAT R ON E.codeEtudiant = R.codeEtudiant
  9       HAVING  AVG(R.note) BETWEEN 12 AND 16
 10      GROUP BY EXTRACT(YEAR FROM E.dateNaissance);

    ANNEE NOMBRE_ETUDIANTS MOYENNE_NOTES NOTE_MAXIMUM NOTE_MINIMUM
--------- ---------------- ------------- ------------ ------------
     1993                1          14,2         14,2         14,2

SQL>
```

Pour la condition de moyenne on va utuliser 'HAVING' :

HAVING AVG(n.note) BETWEEN 12 AND 16 : Filtre les résultats pour inclure uniquement les lignes où la moyenne des notes est comprise entre 12 et 16.

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

```
SQL> SELECT
  2         EXTRACT(YEAR FROM E.dateNaissance) AS annee,
  3          COUNT(E.codeEtudiant) AS nombre_etudiants,
  4          AVG(R.note) AS moyenne_notes,
  5          MAX(R.note) AS note_maximum,
  6          MIN(R.note) AS note_minimum
  7      FROM ETUDIANT E
  8      LEFT JOIN RESULTAT R ON E.codeEtudiant = R.codeEtudiant
  9      GROUP BY EXTRACT(YEAR FROM E.dateNaissance)
 10      ORDER BY moyenne_notes;

    ANNEE NOMBRE_ETUDIANTS MOYENNE_NOTES NOTE_MAXIMUM NOTE_MINIMUM
--------- ---------------- ------------- ------------ ------------
     1993                1          14,2         14,2         14,2
     1992                1          16,8         16,8         16,8
     1990                1          18,5         18,5         18,5
     1994                1          19,3         19,3         19,3
     1991                1            20           20           20

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.

```
SQL> SELECT
  2         EXTRACT(YEAR FROM E.dateNaissance) AS annee,
  3          COUNT(E.codeEtudiant) AS nombre_etudiants,
  4          AVG(R.note) AS moyenne_notes,
  5          MAX(R.note) AS note_maximum,
  6          MIN(R.note) AS note_minimum
  7      FROM ETUDIANT E
  8      LEFT JOIN RESULTAT R ON E.codeEtudiant = R.codeEtudiant
  9      GROUP BY EXTRACT(YEAR FROM E.dateNaissance)
 10      ORDER BY COUNT(E.codeEtudiant)DESC , AVG(R.note) DESC;

    ANNEE NOMBRE_ETUDIANTS MOYENNE_NOTES NOTE_MAXIMUM NOTE_MINIMUM
--------- --------------- ------------- ------------ ------------
     1991               1            20           20           20
     1994               1          19,3         19,3         19,3
     1990               1          18,5         18,5         18,5
     1992               1          16,8         16,8         16,8
     1993               1          14,2         14,2         14,2

SQL>
```

On va l'ordonnée par :

ORDER BY COUNT(E.codeEtudiant) DESC, AVG(R.note) DESC : Trie les résultats en fonction du nombre d'étudiants (en ordre décroissant) et de la moyenne des notes (en ordre décroissant).

13. Donner la liste de noms des enseignants dont le nom contient 'i' à la fin. On préférera que la première lettre du nom soit en majuscule et les autres lettres soient en minuscules.

```
SQL> SELECT
  2      CONCAT(UPPER(SUBSTR(nomEnseignant, 1, 1)), LOWER(SUBSTR(nomEnseignant, 2))) AS nom_formate
  3  FROM ENSEIGNANT
  4  WHERE UPPER(SUBSTR(nomEnseignant, -1)) = 'I';

aucune ligne sÚlectionnÚe

SQL>
```

- **SELECT CONCAT(UPPER(LEFT(nomEnseignant, 1)), LOWER(SUBSTRING(nomEnseignant, 2))) AS Nom_Formate** : Formate le nom pour avoir la première lettre en majuscule et les autres lettres en minuscules.
- **FROM Enseignant** : Spécifie la table **enseignant** à partir de laquelle les données sont extraites.
- **WHERE UPPER(SUBSTR(nomEnseignant, 1)) = 'I'** : Filtre les noms où la dernière lettre, en majuscule (avec la fonction **UPPER**), est égale à 'I'.

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

```
SQL> SELECT nomEnseignant
  2  FROM ENSEIGNANT E
  3  JOIN CHARGE C ON E.codeEnseignant = C.codeEnseignant
  4  GROUP BY E.codeEnseignant, nomEnseignant
  5  HAVING COUNT(C.codeCours) > 2;

aucune ligne sÚlectionnÚe

SQL>
```

HAVING COUNT(C .codeCours) > 2 : Filtre les groupes pour inclure uniquement ceux qui ont plus de deux occurrences, donc les enseignants qui assurent plus de deux cours.

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

```
SQL> SELECT nomEnseignant, specialite
  2  FROM ENSEIGNANT
  3  WHERE specialite IS NOT NULL;

NOMENSEIGNANT        SPECIALITE
-------------------- --------------------
Dumas                Mathématiques
Leroux               Physique
Martin               Informatique
Girard               Chimie
Rousseau             Anglais

SQL>
```

- **SELECT nomEnseignant, specialite** : Sélectionne les colonnes **nom** et **specialite** de la table des enseignants.
- **FROM enseignants** : Spécifie la table **enseignants** à partir de laquelle les données sont extraites.
- **WHERE specialite IS NOT NULL** : Filtre les résultats pour inclure uniquement les lignes où la colonne **specialite** n'est pas nulle, c'est-à-dire où la spécialité est renseignée.

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

```
SQL> SELECT DISTINCT e1.nomEnseignant, e2.nomEnseignant, e1.specialite
  2  FROM Enseignant e1
  3  JOIN Enseignant e2 ON e1.specialite = e2.specialite
  4  WHERE e1.nomEnseignant <> e2.nomEnseignant
  5  ORDER BY e1.specialite, e1.nomEnseignant, e2.nomEnseignant;

aucune ligne sÚlectionnÚe

SQL>
```

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

```
SQL> SELECT DISTINCT e1.nomEnseignant, e2.nomEnseignant, c.intitule AS cours
  2  FROM Enseignant e1
  3  JOIN Charge ch1 ON e1.codeEnseignant = ch1.codeEnseignant
  4  JOIN Charge ch2 ON ch1.codeCours = ch2.codeCours AND ch1.codeEnseignant != ch2.codeEnseignant
  5  JOIN Enseignant e2 ON ch2.codeEnseignant = e2.codeEnseignant
  6  JOIN Cours c ON ch1.codeCours = c.codeCours
  7  ORDER BY c.intitule, 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 distincts de ceux de ETUDIANT. Faire l'union de ETUDIANTS et ETUDIANT.

```
SQL> CREATE TABLE ETUDIANTS AS SELECT * FROM ETUDIANT WHERE 1 = 2;

Table crÚÚe.

SQL>
```

1. **Création d'une nouvelle table nommée ETUDIANTS :** La clause **CREATE TABLE ETUDIANTS** indique à la base de données de créer une nouvelle table appelée ETUDIANTS.
2. **Copie de la structure de la table ETUDIANT :** La partie **AS SELECT * FROM ETUDIANT WHERE 1=2** spécifie la manière dont la nouvelle table ETUDIANTS sera créée. En utilisant **SELECT * FROM ETUDIANT WHERE 1=2**, la base de données demande de copier la structure de la table ETUDIANT (c'est-à-dire ses colonnes et types de données) sans copier les données elles-mêmes.

On va insérer des lignes :

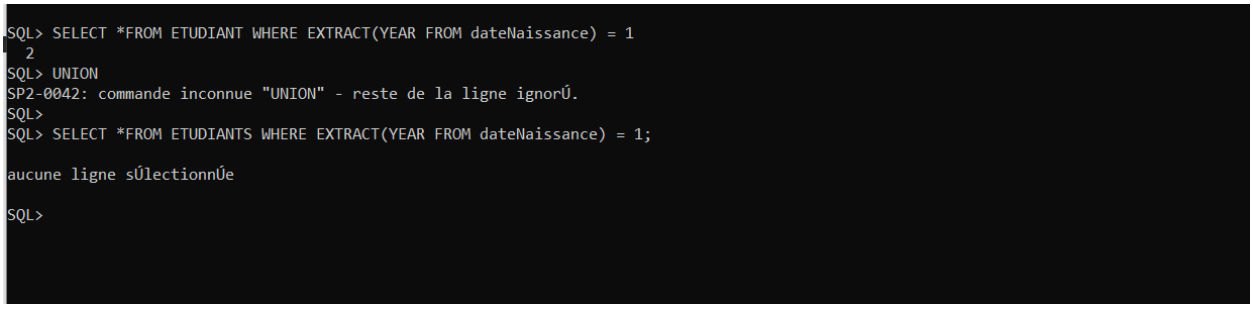
```
SQL> INSERT INTO ETUDIANTS (codeEtudiant, nomEtudiant, prenomEtudiant, dateNaissance, ville)
  2  VALUES
  3       ('E996', 'Izikki', 'Hajar', TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'Agadir');

1 ligne crÚÚe.

SQL>
SQL> INSERT INTO ETUDIANTS (codeEtudiant, nomEtudiant, prenomEtudiant, dateNaissance, ville)
  2  VALUES
  3       ('E997', 'Saket', 'Maryam', TO_DATE('1999-02-15', 'YYYY-MM-DD'), 'Taroudant');

1 ligne crÚÚe.

SQL>
SQL> INSERT INTO ETUDIANTS (codeEtudiant, nomEtudiant, prenomEtudiant, dateNaissance, ville)
  2  VALUES
  3       ('E998', 'Fassi', 'Younesse', TO_DATE('2001-05-20', 'YYYY-MM-DD'), 'Casa');

1 ligne crÚÚe.

SQL>
```

Après on va faire l'union :

```
SQL> SELECT * FROM ETUDIANTS
  2  UNION
  3  SELECT * FROM ETUDIANT;

CODEETUDIA NOMETUDIANT          PRENOMETUDIANT       DATENAIS
---------- -------------------- -------------------- --------
VILLE
--------------------
E001       Dupont               Jean                 15/05/90
Paris

E002       Martin               Sophie               21/08/92
Lyon

E003       Dubois               Pierre               10/03/91
Tiznit


CODEETUDIA NOMETUDIANT          PRENOMETUDIANT       DATENAIS
---------- -------------------- -------------------- --------
VILLE
--------------------
E004       Lefevre              Marie                05/11/93
Taroudant

E005       Moreau               François             28/07/94
Nantes

E996       Izikki               Hajar                01/01/00
Agadir


CODEETUDIA NOMETUDIANT          PRENOMETUDIANT       DATENAIS
---------- -------------------- -------------------- --------
VILLE
--------------------
E997       Saket                Maryam               15/02/99
Taroudant

E998       Fassi                Younesse             20/05/01
Casa


8 ligne(s) sÚlectionnÚe(s).
```

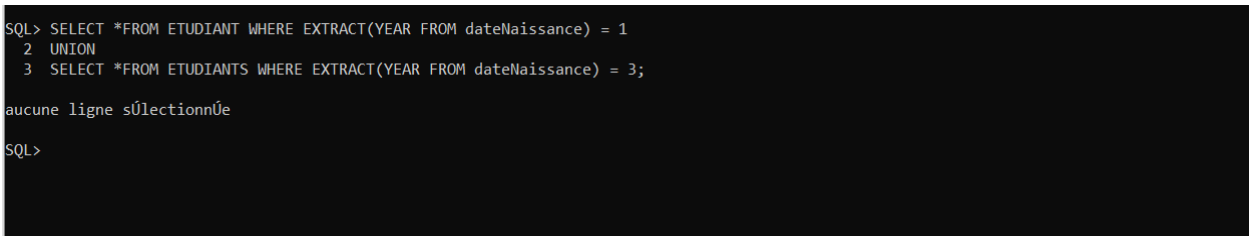
Cela va sélectionner toutes les colonnes de la table **Etudiant** ainsi que de la table **ETUDIANTS** et les concaténer pour retourner un ensemble de résultats sans doublons.

19. Faire l'union de ETUDIANT et de ETUDIANTS pour l'année 1.



- **SELECT * FROM ETUDIANT WHERE EXTRACT(YEAR FROM date_naissance) = 1** : Sélectionne toutes les colonnes de la table **ETUDIANT** où l'année extraite de la colonne **date_naissance** est égale à 1.
- **SELECT * FROM ETUDIANTS WHERE EXTRACT(YEAR FROM date_naissance) = 1** : Sélectionne toutes les colonnes de la table **ETUDIANTS** où l'année extraite de la colonne **date_naissance** est égale à 1.
- **UNION** : Combine les résultats des deux requêtes en une seule liste, en éliminant les doublons.

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



- **SELECT * FROM ETUDIANT WHERE EXTRACT(YEAR FROM date_naissance) = 1** sélectionne toutes les colonnes de la table **ETUDIANT** où l'année extraite de la colonne **date_naissance** est égale à 1 (première année).
- **SELECT * FROM ETUDIANTS WHERE EXTRACT(YEAR FROM date_naissance) = 3** sélectionne toutes les colonnes de la table **ETUDIANTS** où l'année extraite de la colonne **date_naissance** est égale à 3 (troisième année).
- **UNION** combine les résultats des deux requêtes en une seule liste, en éliminant les doublons.

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

```
SQL> SELECT e.codeEtudiant, e.nomEtudiant,
  2         AVG(r.note) AS moyenne_notes,
  3         MIN(r.note) AS note_minimum,
  4         MAX(r.note) AS note_maximum
  5  FROM ETUDIANT e
  6  LEFT JOIN RESULTAT r ON e.codeEtudiant = r.codeEtudiant
  7  GROUP BY e.codeEtudiant, e.nomEtudiant;

CODEETUDIA NOMETUDIANT          MOYENNE_NOTES NOTE_MINIMUM NOTE_MAXIMUM
---------- -------------------- ------------- ------------ ------------
E003       Dubois                         20           20           20
E001       Dupont                       18,5         18,5         18,5
E002       Martin                       16,8         16,8         16,8
E004       Lefevre                      14,2         14,2         14,2
E005       Moreau                       19,3         19,3         19,3

SQL>
```

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

```
SQL> SELECT e.codeEtudiant, e.nomEtudiant,
  2         AVG(r.note) AS moyenne_notes,
  3         MIN(r.note) AS note_minimum,
  4         MAX(r.note) AS note_maximum
  5  FROM ETUDIANT e
  6  LEFT JOIN RESULTAT r ON e.codeEtudiant = r.codeEtudiant
  7  WHERE EXTRACT(YEAR FROM e.dateNaissance) = 1
  8  GROUP BY e.codeEtudiant, e.nomEtudiant;

aucune ligne sÚlectionnÚe

SQL>
```

- **AVG(r.note)** : Calcule la moyenne des notes pour chaque étudiant.
- **MIN(r.note)**: Trouve la note minimale pour chaque étudiant.
- **MAX(r.note)** : Trouve la note maximale pour chaque étudiant.
- **FROM Etudiant e** : Spécifie la table à partir de laquelle les données sont extraites.
- **JOIN Resultat r ON e.etudiant_id = r.etudiant_id** : Jointure entre la table des resultats et la table des étudiants sur l'identifiant de l'étudiant.
- **WHERE EXTRACT(YEAR FROM e.date_naissance) = 1** : Filtre les étudiants en utilisant la fonction **EXTRACT** pour obtenir l'année à partir de la colonne **date_Naissance**.

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

```
SQL> SELECT e.codeEtudiant, e.nomEtudiant,
  2         AVG(r.note) AS moyenne_notes,
  3         MIN(r.note) AS note_minimum,
  4         MAX(r.note) AS note_maximum
  5  FROM ETUDIANT e
  6  LEFT JOIN RESULTAT r ON e.codeEtudiant = r.codeEtudiant
  7  GROUP BY e.codeEtudiant, e.nomEtudiant
  8  HAVING AVG(r.note) > 11;

CODEETUDIA NOMETUDIANT          MOYENNE_NOTES NOTE_MINIMUM NOTE_MAXIMUM
---------- -------------------- ------------- ------------ ------------
E003       Dubois                         20           20           20
E001       Dupont                       18,5         18,5         18,5
E002       Martin                       16,8         16,8         16,8
E004       Lefevre                      14,2         14,2         14,2
E005       Moreau                       19,3         19,3         19,3

SQL>
```

HAVING AVG(note) > 11 : Filtre les résultats pour inclure uniquement les groupes dont la moyenne des notes est supérieure à 11.

24. Donner le nom la moyenne le minimum et le maximum de notes de chaque étudiant de la 2 année et dont la moyenne est supérieure 12.

```
SQL> SELECT e.codeEtudiant, e.nomEtudiant,
  2         AVG(r.note) AS moyenne_notes,
  3         MIN(r.note) AS note_minimum,
  4         MAX(r.note) AS note_maximum
  5  FROM ETUDIANT e
  6  LEFT JOIN RESULTAT r ON e.codeEtudiant = r.codeEtudiant
  7  WHERE EXTRACT(YEAR FROM e.dateNaissance) = 2
  8  GROUP BY e.codeEtudiant, e.nomEtudiant
  9  HAVING AVG(r.note) > 12;

aucune ligne sÚlectionnÚe

SQL>
```

- **SELECT e.codeEtudiant e.nomEtudiant**: Sélectionne le nom de l'étudiant et le code.
- **AVG(r.note) AS Moyenne** : Calcule la moyenne des notes pour chaque étudiant.
- **MIN(r.note) AS Minimum** : Trouve la note minimale pour chaque étudiant.
- **MAX(r.note) AS Maximum** : Trouve la note maximale pour chaque étudiant.
- **FROM Etudiant e** : Spécifie la table à partir de laquelle les données sont extraites.
- **JOIN Resultat e ON r.codeEtudiant = e.codeEtudiant** : Jointure entre la table des Resultats et la table des étudiants sur l'identifiant de l'étudiant.
- **WHERE EXTRACT(YEAR FROM e.dateNaissance) = 2** : Filtre les étudiants de la deuxième année.

- **GROUP BY e.nomEtudiant e .codeEtudiant** : Regroupe les notes par le nom de l'étudiant.
- **HAVING AVG(r.note) > 12** : Filtre les résultats pour inclure uniquement les groupes dont la moyenne des notes est supérieure à 12.

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

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

```
SQL> SELECT e.codeEnseignant, e.nomEnseignant
  2  FROM ENSEIGNANT e
  3  WHERE NOT EXISTS (
  4      SELECT 1
  5      FROM COURS c
  6      WHERE c.intitule = 'Reseaux'
  7      AND NOT EXISTS (
  8          SELECT 1
  9          FROM Charge ch
 10          WHERE ch.codeEnseignant = e.codeEnseignant
 11          AND ch.codeCours = c.codeCours
 12      )
 13  );

CODEENSEIG NOMENSEIGNANT
---------- --------------------
EN001      Dumas
EN002      Leroux
EN003      Martin
EN004      Girard
EN005      Rousseau

SQL>
```

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

```
SQL> SELECT DISTINCT ch.codeEnseignant
  2  FROM CHARGE ch
  3  WHERE NOT EXISTS (
  4      SELECT 1
  5      FROM ENSEIGNANT e
  6      WHERE e.codeEnseignant = ch.codeEnseignant
  7  );

aucune ligne sÚlectionnÚe

SQL>
```

28. Construire une vue qui va contenir les numéros, les noms et les moyennes des étudiants de 1ère année.

```
SQL> CREATE VIEW Vue_Etudiants_1ere_Annee AS
  2  SELECT e.codeEtudiant, e.nomEtudiant, AVG(r.note) AS moyenne_notes
  3  FROM ETUDIANT e
  4  LEFT JOIN RESULTAT r ON e.codeEtudiant = r.codeEtudiant
  5  WHERE EXTRACT(YEAR FROM e.dateNaissance) = 1
  6  GROUP BY e.codeEtudiant, e.nomEtudiant;

Vue crÚÚe.

SQL>
```

CREATE VIEW Vue_Etudiants_1ere_Annee AS : Crée une vue nommée **Vue_Etudiants_1ere_Annee**.

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

```
SQL> SELECT * FROM Vue_Etudiants_1ere_Annee;

aucune ligne sÚlectionnÚe

SQL>
```

Cette requête affichera le contenu de la vue **Vue_Etudiants_1ere _Annee**, qui contient les numéros, les noms et les moyennes des étudiants de première année. Assurez-d'adapter les noms de colonnes et les relations de tables en fonction de votre schéma base de données.