Rapport Bases de données

David HONG

22 décembre 2018

Table des matières

Introduction

Contenus du rapport :

- Le fichier "rapport.pdf" décrivant les TP.
- Un dossier "Images" contenant des dossiers regroupant les screenshots des requêtes des TP classées par TP.
- Des dossiers "TPn" contenant les fichiers script "sql" utilisées.

Les screenshots ont été fait sur le site https://apex.oracle.com. Les screenshots sont en fait des liens ouvrant les images dans le dossier "Images". Si les liens vers les screenshots ne fonctionnent pas il est toujours possible de regarder les screenshots dans le dossier "Images". Les noms des fichiers images sont nommées de la façon suivante : $tp_n.png$ pour les screenshots des requêtes du tp numéro n, nomTable pour les screenshots des contenus des tables, vue_n pour les screenshots des vues. Dans le dossier TP5, les fichier $gen_n.png$ sont les requêtes de la deuxième partie du TP. Dans le dossier TP7, le fichier schemas.png est le schéma E/A de la question 2 du TP7. Les numéros sont classés selon l'ordre des requêtes.

Exemple : Le fichier "tp5_8" correspont au screenshot de la requête numéro 8 du TP5.
Les liens ont été testé dans les salles machine, donc il ne devrait pas y avoir de problème.

Chapitre 1

TP1

1.1 Création d'une base de données sous ORACLE

1.1.1 Première création de table

1) Nous allons créer une table représentant un étudiant dans un fichier etudiants.sql contenant le code ci-dessous.

```
1 CREATE TABLE ETUDIANTS(
2         NUMERO          NUMBER(4)       PRIMARY KEY,
3         NOM             VARCHAR2(25)    NOT NULL,
4         PRENOM          VARCHAR2(25)    NOT NULL,
5         SEXE            CHAR(1)         CHECK(SEXE IN('F','M')),
6         DATENAISSANCE   DATE            NOT NULL,
7         POIDS           NUMBER,
8         ANNEE           NUMBER
9 );
```

Pour créer cette table nous lançons le script etudiants.sql avec la commande `@etudiants_tp1.sql;`.

Un étudiant est caractérisé par :
- un numéro : `NUMERO`
- un nom : `NOM`
- un prénom : `PRENOM`
- un sexe : `SEXE`
- une date de naissance : `DATENAISSANCE`
- un poids : `POIDS`
- une année : `ANNEE`

2) Nous allons vérifier que la table a bien été créée à l'aide deux commandes. La commande `desc ETUDIANTS;` [1] et la commande `select * from ETUDIANTS;`
Affichage de `desc ETUDIANTS;` Image

La commande `select * from ETUDIANTS;` affiche toutes les lignes de notre table. Pour l'instant notre table étant vide, cette commande nous indique `aucune ligne sélectionnée`, nous verrons plus tard cette commande lorque nous aurons insérer des lignes.

3) Dans notre table nous avons défini l'attribut `NUMERO` comme étant la clé primaire. On représente tous les étudiants avec un numéro unique et différent.

1. `DESCRIBE` sous ORACLE

4) Lors de la création de la table nous avons défini plusieurs contraintes.
- NOM : `NOT NULL`
- PRENOM : `NOT NULL`
- SEXE : `CHECK(SEXE IN('F','M'))`
- DATENAISSANCE : `NOT NULL`

Les attributs `NOM`, `PRENOM`, `DATENAISSANCE` ne doivent pas être `NULL`. L'attribut `SEXE` doit être égale à `'F'` ou `'M'`.

5) Après avoir crée notre table, nous allons insérer des nouvelles lignes dans notre table. Pour cela on utilise la commande `INSERT INTO ETUDIANTS VALUES(valeur1,...);`

Exemple nous insérons ces étudiants.

```
1 INSERT INTO ETUDIANTS VALUES(71,'Traifor','Benoît','M','10/12/1978',77,1);
2 INSERT INTO ETUDIANTS VALUES(72,'Génial','Clément','M','10/04/1978',72,1);
3 INSERT INTO ETUDIANTS VALUES(73,'Paris','Adam','M','28/06/1974',72,2);
4 INSERT INTO ETUDIANTS VALUES(74,'Paris','Clémence','F','20/09/1977',72,NULL);
5 INSERT INTO ETUDIANTS VALUES(69,'Saitout','Inès','F','22/11/1969',69,2);
6 INSERT INTO ETUDIANTS VALUES(55,'Serafoub','Izouaf','M','19/09/2013',1,0);
```

Nous allons vérifier le contenu de notre table avec la commande `SELECT * FROM ETUDIANTS;` et voici ce que notre table contient.

NUMERO	NOM	PRENOM	SEXE	DATENAISSANCE	POIDS	ANNEE
71	Traifor	Benoît	M	10/12/78	77	1
72	Génial	Clément	M	10/04/78	72	1
73	Paris	Adam	M	28/06/74	72	2
74	Paris	Clémence	F	20/09/77	72	
69	Saitout	Inès	F	22/11/69	69	2
55	Serafoub	Izouaf	M	19/09/13	1	0

TABLE 1.1 – Etudiants

6) Essayons maintenant d'insérer des lignes qui violent les contraintes définies pour cette table. Essayons d'insérer un étudiant avec numéro `NULL`, deux numéros identiques, un numéro à plus de 4 chiffres, un nom `NULL`, un nom qui dépasse 25 caractères, un sexe différent de 'F' ou 'M'. Voici ce qui se produit.

Insertion d'un étudiant dont le numero est `NULL`. Image

Insertion de deux étudiants ayant le même numéro. Image

Insertion d'une étudiant ayant un numéro à 5 chiffres. Image

Insertion d'un étudiant ayant un nom `NULL` (même résultat pour la date et le prénom) Image

Insertion d'un étudiant ayant un `NOM` à plus de 25 lettres. (même résultat pour le prénom) Image

Insertion d'un étudiant ayant `SEXE` différent de `'M'` ou `'F'`. Image

1.1.2 Base de données avec plusieurs tables

Nous allons créer plusieurs tables dans un fichier ecole_tp1.sql. Nous allons créer des tables qui seront "connectées entre elles". Nous aurons les tables :
- `ELEVES`
- `PROFESSEURS`
- `COURS`

- CHARGE
- RESULTATS
- ACTIVITES
- ACTIVITES_PRATIQUEES

Voici le contenu de notre fichier.

```
DROP TABLE ELEVES CASCADE CONSTRAINTS;
DROP TABLE COURS CASCADE CONSTRAINTS;
DROP TABLE PROFESSEURS CASCADE CONSTRAINTS;
DROP TABLE ACTIVITES CASCADE CONSTRAINTS;
DROP TABLE RESULTATS CASCADE CONSTRAINTS;
DROP TABLE CHARGE CASCADE CONSTRAINTS;
DROP TABLE ACTIVITES_PRATIQUEES CASCADE CONSTRAINTS;

CREATE TABLE ELEVES(
        NUM_ELEVE               NUMBER(4)               ,
        NOM                     VARCHAR2(25)            CONSTRAINT NN_ELEVES_NOM NOT NULL,
        PRENOM                  VARCHAR2(25)            CONSTRAINT NN_ELEVES_PRENOM NOT NULL,
        DATE_NAISSANCE          DATE                    ,
        POIDS                   NUMBER                  ,
        ANNEE                   NUMBER                  ,
        CONSTRAINT PK_ELEVES_NUM PRIMARY KEY (NUM_ELEVE),
        CONSTRAINT CK_ELEVES_POIDS CHECK(POIDS >= 0)    ,
        CONSTRAINT CK_ELEVES_ANNEE CHECK(ANNEE >= 0)
);

CREATE TABLE COURS(
        NUM_COURS               NUMBER(4)               ,
        NOM                     VARCHAR2(25)            CONSTRAINT NN_COURS_NOM NOT NULL,
        NBHEURES                NUMBER                  ,
        ANNEE                   NUMBER                  ,
        CONSTRAINT PK_COURS PRIMARY KEY (NUM_COURS)     ,
        CONSTRAINT CK_COURS_NBHEURES CHECK(NBHEURES > 0),
        CONSTRAINT CK_COURS_ANNEE CHECK(ANNEE >= 0)
);

CREATE TABLE PROFESSEURS(
        NUM_PROF                NUMBER(4)               ,
        NOM                     VARCHAR2(25)            CONSTRAINT NN_PROFESSEURS_NOM NOT NULL,
        SPECIALITE              VARCHAR2(25)            CONSTRAINT NN_PROFESSEURS_SPECIALITE NOT NULL,
        DATE_ENTREE             DATE                    ,
        DER_PROM                DATE                    ,
        SALAIRE_BASE            NUMBER                  ,
        SALAIRE_ACTUEL          NUMBER                  ,
        CONSTRAINT PK_PROFESSEURS_NUM PRIMARY KEY (NUM_PROF),
        CONSTRAINT CK_PROFESSEURS_SALAIRE_BASE CHECK(SALAIRE_BASE >= 0),
        CONSTRAINT CK_PROFESSEURS_SALAIRE_ACTUEL CHECK(SALAIRE_ACTUEL >= 0)
);

CREATE TABLE ACTIVITES(
        NIVEAU  NUMBER          ,
        NOM             VARCHAR2(25)    ,
        EQUIPE  VARCHAR2(25)    ,
        CONSTRAINT PK_ACTIVITES PRIMARY KEY (NIVEAU,NOM)
);

CREATE TABLE RESULTATS(
        NUM_ELEVE               NUMBER(4)               ,
        NUM_COURS               NUMBER(4)               ,
        POINTS                  NUMBER                  ,
        CONSTRAINT FK_ELEVES_NUM_RESULTATS FOREIGN KEY (NUM_ELEVE) REFERENCES ELEVES,
        CONSTRAINT FK_COURS_NUM_RESULTATS FOREIGN KEY (NUM_COURS) REFERENCES COURS
);

CREATE TABLE CHARGE(
        NUM_PROF                NUMBER(4)               ,
        NUM_COURS               NUMBER(4)               ,
        CONSTRAINT FK_COURS_NUM_CHARGE FOREIGN KEY (NUM_COURS) REFERENCES COURS,
        CONSTRAINT FK_PROFESSEURS_NUM_CHARGE FOREIGN KEY (NUM_PROF) REFERENCES PROFESSEURS
);

```

```
66 CREATE TABLE ACTIVITES_PRATIQUEES(
67         NUM_ELEVE                NUMBER(4)                ,
68         NIVEAU                   NUMBER                   ,
69         NOM                      VARCHAR2(25)             CONSTRAINT NN_NOM_ACTIVITES_PRATIQUEES NOT NULL
           ,
70         CONSTRAINT FK_ELEVES_NUM_AP FOREIGN KEY (NUM_ELEVE) REFERENCES ELEVES,
71         CONSTRAINT FK_ACTIVITES_AP FOREIGN KEY (NIVEAU,NOM) REFERENCES ACTIVITES(NIVEAU,NOM)
72 );
```

2) Ici nous utilisons plusieurs fois la commande `DROP TABLE nom_table CASCADE CONSTRAINTS;` au début du fichier. Cette commande supprime les tables ainsi que leurs contraintes car lorsque nous allons lancer la commande `@ecole.sql` si on ne le fait pas, les tables ne vont pas pouvoir être créées.

3) Dans nos tables, nous avons plusieurs clés primaires. Nous utilisons alors la commande `CONSTRAINT nom_contrainte PRIMARY KEY (attribut1,...)`.

4) Nous aurons aussi besoin de clés étrangères. Nous utilisons alors la commande `CONSTRAINT nom_contrainte_FK FOREIGN KEY (attribut1,...) REFERENCES nom_table_de_reference (attribut1,...);` les attributs doivent être égale.

Maintenant que nos tables ont été créées, essayons d'insérer des lignes dans les tables.

```
1  /* INSERTION ELEVES */
2  INSERT INTO ELEVES VALUES(71,'Traifor','Benoît','10/12/1978',77,3);
3  INSERT INTO ELEVES VALUES(72,'Génial','Clément','10/04/1978',72,1);
4  INSERT INTO ELEVES VALUES(73,'Paris','Adam','28/06/1974',72,2);
5  INSERT INTO ELEVES VALUES(74,'Paris','Clémence','20/09/1977',72,NULL);
6  INSERT INTO ELEVES VALUES(69,'Saitout','Inès','22/11/1969',69,2);
7
8  /* INSERTION COURS */
9  INSERT INTO COURS VALUES(1,'Bases de données',20,3);
10 INSERT INTO COURS VALUES(2,'Structures de données',20,2);
11 INSERT INTO COURS VALUES(3,'Programmation Web',10,3);
12 INSERT INTO COURS VALUES(4,'Physique Chimie',20,1);
13 INSERT INTO COURS VALUES(5,'On Fait Rien',10000,0);
14
15 /* INSERTION PROFESSEURS */
16 INSERT INTO PROFESSEURS VALUES(1,'CABANES','INFORMATIQUES',NULL,NULL,50000,50000);
17 INSERT INTO PROFESSEURS VALUES(2,'NOM','RIEN',NULL,NULL,0,0);
18
19 /* INSERTION ACTIVITES */
20 INSERT INTO ACTIVITES VALUES(1,'Football','Sans Nom');
21 INSERT INTO ACTIVITES VALUES(2,'Tennis',NULL);
22 INSERT INTO ACTIVITES VALUES(3,'Football','Équipe de foot');
23
24 /* INSERTION RESULTATS */
25 INSERT INTO RESULTATS VALUES(71,1,18);
26 INSERT INTO RESULTATS VALUES(71,3,17);
27 INSERT INTO RESULTATS VALUES(74,5,20);
28 INSERT INTO RESULTATS VALUES(74,2,8);
29 INSERT INTO RESULTATS VALUES(72,2,10);
30
31 /* INSERTION CHARGE */
32 INSERT INTO CHARGE VALUES(1,1);
33 INSERT INTO CHARGE VALUES(1,5);
34 INSERT INTO CHARGE VALUES(2,5);
35
36 /* INSERTION ACTIVITES PRATIQUEES */
37 INSERT INTO ACTIVITES_PRATIQUEES VALUES(71,1,'Football');
38 INSERT INTO ACTIVITES_PRATIQUEES VALUES(72,3,'Football');
39 INSERT INTO ACTIVITES_PRATIQUEES VALUES(74,2,'Tennis');
```

Nos tables ont bien été ajoutées. Les contraintes sont bien respecté.

Chapitre 2

TP2

2.1 Modification d'une base de données sous ORACLE

2.1.1 Modification de contraintes

1) Voici une table représentant un étudiant.

```
CREATE TABLE ETUDIANTS(
    NUMERO          NUMBER(4)       NOT NULL,
    NOM             VARCHAR2(25)    NOT NULL,
    PRENOM          VARCHAR2(25)    NOT NULL,
    SEXE            CHAR(1)         CHECK(SEXE IN('F','M')),
    DATENAISSANCE   DATE            NOT NULL,
    POIDS           NUMBER          ,
    ANNEE           NUMBER          ,
    CONSTRAINT PK_ETUDIANTS PRIMARY KEY (NUMERO)
);
```

2) On utilise la commande `select constraint_name from user_constraints where table_name='ETUDIANTS';`. Voici le résultat contraintes 1.

Cette commande liste donc toutes les contraintes ainsi que leurs noms, étant donné que l'on a plusieurs contraintes mais que nous ne les avons pas nommée, ils ont des noms par défaut. Nous allons corriger le code précédent en nommant les contraintes. Voici le résultat contraintes 2.

```
CREATE TABLE ETUDIANTS(
        NUMERO          NUMBER(4)       ,
        NOM             VARCHAR2(25)    CONSTRAINT CONTRAINTE1 NOT NULL,
        PRENOM          VARCHAR2(25)    CONSTRAINT CONTRAINTE2 NOT NULL,
        SEXE            CHAR(1)         CONSTRAINT CONTRAINTE3 CHECK(SEXE IN ('F','M')),
        DATENAISSANCE   DATE            CONSTRAINT CONTRAINTE4 NOT NULL,
        POIDS           NUMBER          ,
        ANNEE           NUMBER          ,
        CONSTRAINT PK_ETUDIANTS PRIMARY KEY (NUMERO)
);
```

3) Nous allons ajouter deux nouvelles contraintes.

- L'année doit être égale à 1 ou 2
- Le poids doit être supérieur à 30kg et inférieur à 200kg

Pour cela on utilise la commande `ALTER TABLE nom_table ADD CONSTRAINT nom_contrainte CHECK(contrainte);`

```
ALTER TABLE ETUDIANTS ADD CONSTRAINT CK_POIDS CHECK(POIDS > 30 AND POIDS < 200);
ALTER TABLE ETUDIANTS ADD CONSTRAINT CK_ANNEE CHECK(ANNEE=1 OR ANNEE=2);
```

On vérifie si les contraintes ont été rajouté. La commande `select constraint_name from user_constraints where table_name='ETUDIANTS';` nous affiche ceci contraintes 3

4) On renomme les contraintes avec la commande `ALTER TABLE nom_table RENAME CONSTRAINT nom_contrainte1 TO nom_contrainte2;`

```
ALTER TABLE ETUDIANTS RENAME CONSTRAINT CONTRAINTE1 TO NN_NOM;
ALTER TABLE ETUDIANTS RENAME CONSTRAINT CONTRAINTE2 TO NN_PRENOM;
ALTER TABLE ETUDIANTS RENAME CONSTRAINT CONTRAINTE3 TO CK_SEXE;
ALTER TABLE ETUDIANTS RENAME CONSTRAINT CONTRAINTE4 TO NN_DATENAISSANCE;
```

Vérifions les noms de nos contraintes contraintes 4.

2.1.2 Manipulation de la BD École

1) Fichier ecole_tp2.sql.

2) Pour ajouter une contrainte de clé étrangère on utilise la commande `ALTER TABLE nom_table ADD CONSTRAINT nom_contrainte FOREIGN KEY (colonne) REFERENCES nom_table(colonne)`. Ajoutons dans le fichier `ecole.sql` les clés étrangères via ces commandes.

```
ALTER TABLE CHARGE ADD CONSTRAINT FK_PROFESSEURS_NUM_CHARGE FOREIGN KEY (NUM_PROF) REFERENCES
    PROFESSEURS(NUM_PROF) ;
ALTER TABLE CHARGE ADD CONSTRAINT FK_COURS_NUM_CHARGE FOREIGN KEY (NUM_COURS) REFERENCES COURS(
    NUM_COURS) ;
ALTER TABLE RESULTATS ADD CONSTRAINT FK_ELEVES_NUM_RESULTATS FOREIGN KEY (NUM_ELEVE) REFERENCES ELEVES(
    NUM_ELEVE) ;
ALTER TABLE RESULTATS ADD CONSTRAINT FK_COURS_NUM_RESULTATS FOREIGN KEY (NUM_COURS) REFERENCES COURS(
    NUM_COURS) ;
ALTER TABLE ACTIVITES_PRATIQUEES ADD CONSTRAINT PK_ELEVES_NUM_AP FOREIGN KEY (NUM_ELEVE) REFERENCES
    ELEVES(NUM_ELEVE) ;
ALTER TABLE ACTIVITES_PRATIQUEES ADD CONSTRAINT FK_ACTIVITE_AP FOREIGN KEY (NIVEAU,NOM) REFERENCES
    ACTIVITES(NIVEAU,NOM) ;
```

3) Affichage de la commande `DESC ELEVES;` Image

4) Pour ajouter des attributs dans une table déjà existante on peut utiliser la commande `ALTER TABLE nom_table ADD (nom_colonnes);`. Nous allons ajouter les attributs `CodePostal` et `Ville` dans la table des élèves.

```
ALTER TABLE ELEVES ADD (
        CODEPOSTAL              NUMBER(5)               ,
        VILLE                   VARCHAR2(20)
);
```

5) Pour mettre à jour les données d'une table, on peut utiliser la commande `UPDATE nom_table SET colonne = valeur WHERE condition;`

```
UPDATE ELEVES SET CODEPOSTAL = 75013, VILLE = 'paris' WHERE NUM_ELEVE=1;
UPDATE ELEVES SET CODEPOSTAL = 93800, VILLE = 'EPINAY / seine' WHERE NUM_ELEVE=2;
UPDATE ELEVES SET CODEPOSTAL = 93430, VILLE = 'EPINAY_SUR_SEINE' WHERE NUM_ELEVE=5;
UPDATE ELEVES SET CODEPOSTAL = 91000, VILLE = 'EPINAY / ORGE' WHERE NUM_ELEVE=7;
```

6) Nous allons créer une nouvelle table `AGGLOMERATION`.

```
CREATE TABLE AGGLOMERATION(
        CP              NUMBER(5),
        Ville           VARCHAR2(25)
);
```

7) Ajout des contraintes dans la table `AGGLOMERATION`.

— `CP` et `Ville` doivent être la clé primaire de `AGLOMERATION`

— `Ville` doit être en majuscule (fonction `UPPER`)

```
1 ALTER TABLE AGGLOMERATION ADD CONSTRAINT PK_AGGLOMERATION PRIMARY KEY (CP, VILLE);
2 ALTER TABLE AGGLOMERATION ADD CONSTRAINT CK_VILLE_AGGLOMERATION CHECK(VILLE=UPPER(VILLE));
```

8) La commande `INSERT INTO AGGLOMERATION VALUES(93430,'Villetaneuse');` ne fonctionnera pas car la ville n'est pas en majuscules.

```
1 INSERT INTO AGGLOMERATION VALUES(75001, 'PARIS');
2 INSERT INTO AGGLOMERATION VALUES(75013, 'PARIS');
3 INSERT INTO AGGLOMERATION VALUES(93800, 'EPINAY SUR SEINE');
4 INSERT INTO AGGLOMERATION VALUES(93430,UPPER('VILLETANEUSE'));
5 INSERT INTO AGGLOMERATION VALUES(91000, 'EPINAY SUR ORGE');
6 INSERT INTO AGGLOMERATION VALUES(93800, 'EPINAY / SEINE');
```

CP	VILLE
75001	PARIS
75013	PARIS
91000	EPINAY SUR ORGE
93430	VILLETANEUSE
93800	EPINAY / SEINE
9380	EPINAY SUR SEINE

TABLE 2.1 – Agglomeration

9) On va mettre à jour les noms des villes selon le code postal. Pour cela on utilise la commande `UPDATE Table1 SET AttributAMettreAJour = (SELECT Attribut FROM Table2 WHERE Condition);`

```
1 UPDATE ELEVES SET VILLE = (SELECT VILLE FROM AGGLOMERATION WHERE CODEPOSTAL = CP AND ROWNUM = 1);
```

On utilise `ROWNUM = 1` car dans notre table `AGGLOMERATION` il y a deux code postal avec deux nom de ville différentes ('EPINAY / SEINE' et 'EPINAY SUR SEINE'). Si c'est le cas on va choisir le premier rencontré.

Chapitre 3

TP3

3.1 Fonctions ORACLE

3.1.1 Exploration de quelques fonctions ORACLE

1) Nous allons nous intéresser à ces commandes ci-dessous :

- `SELECT RPAD('Soleil',17,'bla') "RPAD exemple" FROM DUAL;` cette commande prend le mot 'Soleil' et le concatène à la fin la séquence de mot 'bla' jusqu'à ce que la taille du mot soit égale à 17. (cela affiche donc 'Soleilblablablabl') Image
- `SELECT LPAD('Master 2 EID',15,'*.') "LPAD exemple" FROM DUAL;` cette commande prend le mot 'Master 2 EID' et le concatène à partir du début la séquence de mot '*.' jusqu'à ce que la taille du mot soit égale à 15. (cela affiche '*.*Master 2 EID') Image
- `SELECT SUBSTR('DESS EID',6,3) "SUBSTR exemple" FROM DUAL;` cette commande prend le mot 'DESS EID' et affiche les trois lettres du mot à partir de la 6^{ième} caractère. (cela affiche 'EID') Image
- `SELECT SUBSTR('ABCDEFGHIJ',-5,4) "SUBSTR exemple" FROM DUAL;` cette commande prend le mot 'ABCDEFGHIJ' et affiche les quatre lettres du mot à partir de la 5^{ième} caractère en partant de la fin du mot. (cela affiche 'FGHI') Image
- `SELECT TO_CHAR(SYSDATE, 'MM-DD-YYYY HH24:MI:SS') "Now" FROM DUAL;` cette commande affiche la date sous forme `MM-DD-YYYY` et l'heure sous forme `heures:minutes:secondes`. Image
- `SELECT LENGTH('WEB WAREHOUSE') "Longueur en caractères" FROM DUAL;` cette commande affiche la longueur du mot 'WEB WAREHOUSE'. (cela affiche 13) Image
- `SELECT ROUND(17.0958,1) "ROUND exemple" FROM DUAL;` cette commande affiche l'arrondi supérieur du nombre 17.0958 à 1 chiffre après la virgule. (cela affiche 17.1) Image
- `SELECT ROUND(17.58,2) "ROUND exemple" FROM DUAL;` cette commande affiche l'arrondi supérieur du nombre 17.58 à 2 chiffres après la virgule. (cela affiche 17.58) Image
- `SELECT TRUNC(1958.0917,1) "TRUNC exemple" FROM DUAL;` cette commande affiche le nombre 1958.0917 à 1 chiffre après la virgule. (cela affiche 1958) Image
- `SELECT TRUNC(1958.0917,2) "TRUNC exemple" FROM DUAL;` cette commande affiche le nombre 1958.0917 à 2 chiffres après la virgule. (cela affiche 1958.09) Image
- `SELECT ROUND(TO_DATE('SEP-17-2009'), 'YEAR') "New Year" FROM DUAL;` cette commande affiche le premier jour de l'année suivante si le nombre de jours restant pour l'année suivante est plus petit que le nombre de jours passés, affiche le premier jour de l'année précédente sinon. (exemple si on est le 01/02/2010 cela affiche 01/01/2010 or si on est le 01/12/2010 cela affiche 01/01/2011) Image
- `SELECT SYSDATE FROM DUAL;` cette commande affiche la date sous forme `jours/mois/années` Image.
- `SELECT EXTRACT(YEAR FROM SYSDATE) FROM DUAL;` cette commande affiche l'année. Image
- `SELECT ADD_MONTHS(SYSDATE,7) FROM DUAL;` cette commande affiche la date actuelle ou l'on a ajouté 7 mois. (exemple si on est le 01/01/2010, cela affiche 01/08/2010) Image
- `SELECT TRUNC(MONTHS_BETWEEN(SYSDATE, TO_DATE('JUN-19-2001'))) AS AGEBB FROM DUAL;` cette commande affiche le nombre de mois qu'il y a entre le 19/06/2001 et la date actuelle Image.
- `SELECT TO_NUMBER(TO_CHAR(SYSDATE, 'YYYY')) FROM DUAL;` cette commande commence par convertir l'année actuelle (`DATE`) en chaîne de caractères (`CHAR`) puis le convertis en un nombre (`NUMBER`). (cela affiche 2018) Image

Remarque : `DUAL` est un objet permettant de faire `SELECT ... FROM` sans avoir crée de table.

2) On change le format des dates en utilisant la commande `ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MM-YYYY';` le format de date par défaut étant 'YYYY-MM-DD'. Un message nous indique que la session a été modifiée.

3.1.2 Exemple sur de vrai table

1) Commençons par créer une table `ETUDIANTS`.

```
CREATE TABLE ETUDIANTS(
        NUMERO          NUMBER(4)       NOT NULL,
        NOM             VARCHAR2(25)    NOT NULL,
        PRENOM          VARCHAR2(25)    NOT NULL,
        SEXE            CHAR(1)         CHECK (SEXE IN ('M','F')),
        DATENAISSANCE   DATE            NOT NULL,
        POIDS           NUMBER          ,
        ANNEE           NUMBER          ,
        CONSTRAINT PK_ETUDIANTS PRIMARY KEY (NUMERO)
);
```

2) Insérons des lignes.

```
INSERT INTO ETUDIANTS VALUES(71,'Traifor','Benoît','M','10/12/1978',77,1);
INSERT INTO ETUDIANTS VALUES(72,'Génial','Clément','M','10/04/1978',72,1);
INSERT INTO ETUDIANTS VALUES(73,'Paris','Adam','M','28/06/1974',72,2);
INSERT INTO ETUDIANTS (NUMERO,NOM,PRENOM,SEXE,DATENAISSANCE,POIDS) VALUES(74,'Parees','Clémence','F','
    20/09/1977',72);
INSERT INTO ETUDIANTS VALUES(69,'Saitout','Inès','F','22/11/1969',69,2);
INSERT INTO ETUDIANTS VALUES(55,'Seratoub','Izouaf','M','19/09/2013',81,1);
```

Voici notre table.

NUMERO	NOM	PRENOM	SEXE	DATENAISSANCE	POIDS	ANNEE
71	Traifor	Benoît	M	10/12/1978	77	1
72	Génial	Clément	M	10/04/1978	72	1
73	Paris	Adam	M	28/06/1974	72	2
74	Parees	Clémence	F	20/09/1977	72	
69	Saitout	Inès	F	22/11/1969	69	2
55	Seratoub	Izouaf	M	19/09/2013	81	1

TABLE 3.1 – Etudiants

3) Nous allons tester plusieurs requêtes

— Cette requête affiche dans une colonne que l'on nomme `ANETUDE` `Première` si `ANNEE = 1`, `Seconde` si `ANNEE = 2`, `Valeur différente de 1 et de 2 !!` si `ANNEE != 1 OR ANNEE != 2` Image.

```
SELECT DECODE(ANNEE,1,'Première',2,'Seconde','Valeur_différente_de_1_et_de_2_!!') AS ANETUDE FROM
    ETUDIANTS;
```

— Cette requête affiche le nom de tous les étudiants en majuscule. Image.

```
SELECT UPPER(NOM) FROM ETUDIANTS;
```

— Cette requête affiche le nom de tous les étudiants en minuscule. Image.

```
SELECT LOWER(NOM) FROM ETUDIANTS;
```

— `SELECT NVL(ANNEE,'Valeur NON renseignée') AS AN_ETUDE FROM ETUDIANTS;` cette commande affiche une erreur "ORA-01722 : Nombre non valide".

La fonction `NVL` nous affiche une erreur car `ANNEE` est un nombre alors que 'Valeur NON renseignée' est une chaîne de caractères, or cette fonction met chaque valeur `ANNEE` de la table ayant `NULL` comme valeur par 'Valeur NON renseignée'. On essaye donc ici de mettre une chaîne de caractères dans un nombre. Image.

```
SELECT NVL(TO_CHAR(ANNEE), 'Valeur_NON_renseignée') AS ANETUDE FROM ETUDIANTS;
```

4) Gestion de l'affichage. On utilise les commandes ci-dessous (ne fonctionne pas sur https://apex.oracle.com) :

```
COL attribut FORMAT format
TITLE 'Un titre'
SET PAGES n
SET LINES m
```

5) Nous souhaitons interroger notre BD.

1. Affiche le nom et prénom dans une seul colonne Image.

```
SELECT NOM||' '||PRENOM AS NOM_PRENOM FROM ETUDIANTS;
```

2. Affiche la première lettre du prénom en majuscule suivit d'un '. ' puis le nom en majuscule dans une seul colonne Image.

```
SELECT UPPER(SUBSTR(PRENOM,1,1))||'. '||UPPER(NOM) AS PN FROM ETUDIANTS WHERE SEXE = 'M';
```

3. Affiche le nom et la date de naissance des étudiants dont le nom se prononce 'Paris' Image.

```
SELECT NOM,DATENAISSANCE FROM ETUDIANTS WHERE SOUNDEX(NOM) = SOUNDEX('Paris');
```

4. Affiche le nom et la date de naissance des étudiants ayant un prénom qui commence par la lettre 'I' Image.

```
SELECT NOM,DATENAISSANCE FROM ETUDIANTS WHERE PRENOM LIKE 'I%';
```

5. Affiche une description de l'étudiant. (affiche le nom, prénom, son sexe, sa date de naissance, son poids et son année) Image.

```
SELECT UPPER(NOM)||' '||PRENOM||' est '||DECODE(SEXE, 'M', 'un garçon', 'F', 'une fille')||' né
    le '||DATENAISSANCE||' pèse '||POIDS||' kg et est en '||DECODE(ANNEE, 1, 'Première année', 2,
     'Seconde années') AS DESCRIPTION FROM ETUDIANTS WHERE ANNEE BETWEEN 1 AND 2;
```

6. Affiche l'âge de l'étudiant Image.

```
SELECT UPPER(NOM)||' à '||TO_CHAR(EXTRACT(YEAR FROM SYSDATE)-EXTRACT(YEAR FROM DATENAISSANCE)) ||
    ' ans' AS AGE FROM ETUDIANTS;
```

Chapitre 4

TP4

4.1 SQL Simple, Tri et regroupements

4.1.1 Table employés

1) Voici une table représentant des employés.

```
CREATE TABLE EMPLOYE(
        NUM_EMP         NUMBER(4)                   ,
        NOM_EMP         VARCHAR(25)                 CONSTRAINT NN_NOM_EMP NOT NULL,
        DATE_EMB        DATE                        CONSTRAINT NN_DATE_EMB NOT NULL,
        DATE_SORTIE     DATE                        ,
        CONSTRAINT PK_EMPLOYE PRIMARY KEY  (NUM_EMP)
);
```

On va insérer des lignes dans notre table.

```
INSERT INTO EMPLOYE (NUM_EMP,NOM_EMP,DATE_EMB) VALUES(9007,'CHEVALIER','01/01/1996');
INSERT INTO EMPLOYE (NUM_EMP,NOM_EMP,DATE_EMB) VALUES(9701,'LEROY','17/09/1997');
INSERT INTO EMPLOYE (NUM_EMP,NOM_EMP,DATE_EMB) VALUES(9703,'LAMI','17/09/1997');
INSERT INTO EMPLOYE (NUM_EMP,NOM_EMP,DATE_EMB) VALUES(9801,'SULTAN','20/03/1998');
INSERT INTO EMPLOYE (NUM_EMP,NOM_EMP,DATE_EMB) VALUES(9802,'CLEMENCE','16/10/1998');
INSERT INTO EMPLOYE (NUM_EMP,NOM_EMP,DATE_EMB) VALUES(9803,'CAVALIER','22/11/1998');
INSERT INTO EMPLOYE (NUM_EMP,NOM_EMP,DATE_EMB) VALUES(9901,'ALEXANDRE','21/02/1999');
```

NUM_EMP	NOM_EMP	DATE_EMB	DATE_SORTIE
9007	CHEVALIER	01/01/1969	
9701	LEROY	17/09/1997	
9703	LAMI	17/09/1997	
9801	SULTAN	20/03/1998	
9802	CLÉMENCE	16/10/1998	
9803	CAVALIER	22/11/1998	
9901	ALEXANDRE	21/02/1999	

Table 4.1 – Employé

2) On va essayer de les afficher.

1. Affiche tous les employés Image.

```
SELECT * FROM EMPLOYE;
```

2. Affiche le nom de tous les employés Image.

```
SELECT NOM_EMP AS NOM FROM EMPLOYE;
```

3. Affiche le nom des employés embauchés à partir du 1er janvier 1999 Image.

```
SELECT NOM_EMP AS NOM FROM EMPLOYE WHERE DATE_EMB >= '01/01/1999';
```

4. Affiche le numéro et le nom des employés qui ont leurs nom qui commence par la lettre 'C' Image.

```
SELECT NUM_EMP,NOM_EMP AS NOM FROM EMPLOYE WHERE NOM_EMP LIKE 'C%';
```

5. Affiche le nom des employés triés par ordre décroissant sur les noms Image.

```
SELECT NOM_EMP AS NOM FROM EMPLOYE ORDER BY NOM_EMP DESC;
```

6. Affiche le nombre d'employés embauchés chaque année Image.

```
SELECT COUNT(DATE_EMB) AS NB_EMPLOYE,EXTRACT(YEAR FROM DATE_EMB) AS ANNEE FROM EMPLOYE GROUP BY
    EXTRACT(YEAR FROM DATE_EMB);
```

7. Affiche le nombre d'employés embauchés chaque année ayant un nom de plus de 5 lettres Image.

```
SELECT COUNT(DATE_EMB) AS NB_EMPLOYE,EXTRACT(YEAR FROM DATE_EMB) AS ANNEE FROM EMPLOYE WHERE
    LENGTH(NOM_EMP) > 5 GROUP BY EXTRACT(YEAR FROM DATE_EMB);
```

8. Affiche le nombre d'employés embauchés chaque année ayant un nom commençant par un 'L' ou 'C' en ne gardant que les années avec au moins deux employés Image.

```
SELECT COUNT(EXTRACT(YEAR FROM DATE_EMB)) AS NB_EMPLOYE,EXTRACT(YEAR FROM DATE_EMB) AS ANNEE FROM
    EMPLOYE WHERE (NOM_EMP LIKE 'L%' OR NOM_EMP LIKE 'C%') GROUP BY EXTRACT(YEAR FROM DATE_EMB)
    HAVING COUNT(EXTRACT(YEAR FROM DATE_EMB)) >= 2;
```

4.2 Table postes

1) Nous allons recréer une nouvelle table employé.

```
CREATE TABLE EMPLOYE(
        NumEmp          NUMBER(4)                   ,
        Poste           VARCHAR2(25)                CONSTRAINT NN_Poste NOT NULL,
        Salaire         NUMBER                      CONSTRAINT CK_Salaire CHECK(Salaire >= 0),
        NumServ         VARCHAR2(2)                 CONSTRAINT NN_NumServ NOT NULL,
        DateDeb         DATE                        CONSTRAINT NN_DateDeb NOT NULL,
        DateFin         DATE                        ,
        CONSTRAINT PK_EMPLOYE PRIMARY KEY (NumEmp)
);
```

On insert les lignes suivantes :

```
INSERT INTO EMPLOYE (NumEmp, Poste, Salaire, NumServ, DateDeb) VALUES(9701, 'PRESIDENT',5800, 'S2', '
    17/09/1997');
INSERT INTO EMPLOYE (NumEmp, Poste, Salaire, NumServ, DateDeb, DateFin) VALUES(9703, 'SECRETAIRE',950, 'S1', '
    17/09/1997', '31/12/1998');
INSERT INTO EMPLOYE (NumEmp, Poste, Salaire, NumServ, DateDeb) VALUES(9703, 'SECRETAIRE',1200, 'S1', '
    01/01/1999');
INSERT INTO EMPLOYE (NumEmp, Poste, Salaire, NumServ, DateDeb, DateFin) VALUES(9801, 'DIRECTEUR',5300, 'S1', '
    07/07/1997', '31/12/1998');
INSERT INTO EMPLOYE (NumEmp, Poste, Salaire, NumServ, DateDeb) VALUES(9801, 'DIRECTEUR',3200, 'S5', '
    20/03/1998');
INSERT INTO EMPLOYE (NumEmp, Poste, Salaire, NumServ, DateDeb) VALUES(9802, 'DIRECTEUR',3500, 'S2', '
    16/10/1998');
INSERT INTO EMPLOYE (NumEmp, Poste, Salaire, NumServ, DateDeb) VALUES(9803, 'INGENIEUR',2600, 'S4', '
    22/11/1998');
INSERT INTO EMPLOYE (NumEmp, Poste, Salaire, NumServ, DateDeb) VALUES(9901, 'DIRECTEUR',3000, 'S3', '
    21/02/1999');
```

NumEmp	Poste	Salaire	NumServ	DateDeb	DateFin
9701	PRESIDENT	5800	S2	17/09/1997	
9703	SECRETAIRE	950	S1	17/09/1997	31/12/1998
9703	SECRETAIRE	1200	S1	01/01/1999	
9801	DIRECTEUR	5300	S1	07/07/1997	31/12/1998
9801	DIRECTEUR	3200	S5	20/03/1998	
9802	DIRECTEUR	3500	S2	16/10/1998	
9803	INGENIEUR	2600	S4	22/11/1998	
9901	DIRECTEUR	3000	S3	21/02/1999	

TABLE 4.2 – Employé

2) On va essayer de les afficher.

1. Affiche les noms des postes (sans doublons) Image.

```
SELECT DISTINCT Poste FROM EMPLOYE;
```

2. Affiche les postes occupés dont le salaire de l'employé est supérieur ou égal à 3000 Image.

```
SELECT Poste, Salaire FROM EMPLOYE WHERE Salaire >= 3000;
```

3. Affiche les postes occupés, triés par ordre décroissant et salaire par ordre croissant Image.

```
SELECT Poste, Salaire FROM EMPLOYE WHERE Salaire >= 3000 ORDER BY Salaire DESC;
```

4. Affiche le salaire le plus bas Image.

```
SELECT MIN(Salaire) AS SALAIRE_MIN FROM EMPLOYE;
```

5. Affiche la moyenne des salaires Image.

```
SELECT AVG(Salaire) AS Moyenne_Salaire FROM EMPLOYE;
```

6. Affiche la moyenne des salaires par postes Image.

```
SELECT AVG(Salaire) AS Moyenne_Salaire, Poste FROM EMPLOYE GROUP BY Poste;
```

7. Affiche le nombre de salariés avec un salaire > 3000 Image.

```
SELECT COUNT(Salaire) AS Nombre_Salaire_Sup_3000 FROM EMPLOYE WHERE Salaire > 3000;
```

8. Affiche la moyenne des salaires actuels pour chaque service Image.

```
SELECT AVG(Salaire) AS Moyenne_Salaire, NumServ FROM EMPLOYE GROUP BY NumServ;
```

9. Affiche la moyenne des salaires pour chaque poste avec au moins 2 employés Image.

```
SELECT AVG(Salaire) AS Moyenne_Salaire, Poste FROM EMPLOYE GROUP BY Poste HAVING COUNT(Poste) >=
    2;
```

4.2.1 Table Etudiants

NUMERO	NOM	PRENOM	SEXE	DATENAISSANCE	POIDS	ANNEE
71	Traifor	Benoît	M	10/12/1978	77	1
72	Génial	Clément	M	10/04/1978	72	1
73	Paris	Adam	M	28/06/1974	72	2
74	Parees	Clémence	F	20/09/1977	72	
69	Saitout	Inès	F	22/11/1969	69	2
55	Seratoub	Izouaf	M	19/09/2013	81	1

TABLE 4.3 – Etudiants

Considérons notre table ci-dessous notre table étudiants.

1. Affiche la moyenne des poids par sexe Image.

```
1 SELECT AVG(POIDS) AS MOYENNE_POIDS,SEXE FROM ETUDIANTS GROUP BY SEXE;
```

2. Affiche la moyenne des poids par sexe et par tranche d'âge Image.

```
1 SELECT AVG(POIDS)AS MOYENNE_POIDS,EXTRACT(YEAR FROM SYSDATE)-EXTRACT(YEAR FROM DATENAISSANCE) AS
      AGE,SEXE FROM ETUDIANTS GROUP BY SEXE,EXTRACT(YEAR FROM DATENAISSANCE) ;
```

3. Affiche la moyenne des poids par année, par sexe et par tranche d'âge Image.

```
1 SELECT AVG(POIDS) AS MOYENNE_POIDS,ANNEE,SEXE,EXTRACT(YEAR FROM SYSDATE)-EXTRACT(YEAR FROM
      DATENAISSANCE) AS AGE FROM ETUDIANTS GROUP BY ANNEE,SEXE,EXTRACT(YEAR FROM SYSDATE)-EXTRACT(
      YEAR FROM DATENAISSANCE) ;
```

4. Affiche la moyenne des poids par sexe, par année et par tranche d'âge Image.

```
1 SELECT AVG(POIDS) AS MOYENNE_POIDS,SEXE,ANNEE,EXTRACT(YEAR FROM SYSDATE)-EXTRACT(YEAR FROM
      DATENAISSANCE) AS AGE FROM ETUDIANTS GROUP BY SEXE,ANNEE,EXTRACT(YEAR FROM SYSDATE)-EXTRACT(
      YEAR FROM DATENAISSANCE) ;
```

Chapitre 5

TP5

5.1 SQL : Jointures

5.1.1 Gestion d'un café

1) Liste des fichiers :

- creatcafe.sql
- insertcafe.sql
- requetes_tp5.sql

Voici les contenus de nos tables (résultat de la requête 1) :

- La table `LESTABLES` Image
- La table `SERVEUR` Image
- La table `CONSOMMATION` Image
- La table `FACTURE` Image
- La table `COMPREND` Image

La liste des commandes, les numéros de lignes correspondent avec le numéros des requêtes de l'annexe A Appendice A

```
2  SELECT NBPLACE FROM LESTABLES WHERE NUMTABLE = 4;
3  SELECT NUMCONS,LIBCONS,PRIXCONS FROM CONSOMMATION WHERE PRIXCONS > 1;
4  SELECT NUMSERVEUR,NOMSERVEUR,VILLESERVEUR FROM SERVEUR WHERE VILLESERVEUR = 'BELFORT' OR VILLESERVEUR =
       'DELLE';
5  SELECT NUMFACTURE,NUMTABLE FROM FACTURE WHERE NUMSERVEUR = 52 AND DATEFACTURE = '02/02/2010';
6  SELECT NUMCONS,QTE FROM COMPREND WHERE NUMFACTURE = 1203;
7  SELECT DISTINCT NUMCONS FROM COMPREND WHERE NUMFACTURE = 1200 OR NUMFACTURE = 1201;
8  SELECT NOMSERVEUR,DATENSERVEUR FROM SERVEUR WHERE EXTRACT(YEAR FROM DATENSERVEUR) = '1976';
9  SELECT NUMCONS,LIBCONS,PRIXCONS FROM CONSOMMATION WHERE LIBCONS LIKE 'Biere%';
10 SELECT * FROM FACTURE WHERE DATEFACTURE > '01/02/2010';
11 SELECT NOMSERVEUR FROM SERVEUR WHERE NOMSERVEUR LIKE '_i%';
12 SELECT NOMSERVEUR FROM SERVEUR WHERE NOMSERVEUR LIKE 'P%';
13 SELECT NOMSERVEUR FROM SERVEUR ORDER BY VILLESERVEUR;
14 SELECT LIBCONS,NUMCONS,PRIXCONS FROM CONSOMMATION ORDER BY LIBCONS;
15 SELECT DISTINCT VILLESERVEUR FROM SERVEUR;
16 SELECT COUNT(*) AS NOMBRE_TABLES FROM LESTABLES;
17 SELECT SUM(NBPLACE) AS NOMBRE_PLACES FROM LESTABLES;
18 SELECT NUMSERVEUR,COUNT(*) AS NB_FACTURE FROM FACTURE GROUP BY NUMSERVEUR;
19 SELECT DATEFACTURE,COUNT(*) AS NB_FACTURE FROM FACTURE GROUP BY DATEFACTURE;
20 SELECT NUMSERVEUR,COUNT(*) AS NB_FACTURE FROM FACTURE GROUP BY NUMSERVEUR HAVING COUNT(NUMSERVEUR) > 3;
21 SELECT AVG(PRIXCONS) AS MOYENNE_PRIX FROM CONSOMMATION;
22 SELECT AVG(PRIXCONS) AS MOYENNE_PRIX FROM CONSOMMATION WHERE LIBCONS LIKE 'Cafe%';
23 SELECT NUMCONS,AVG(QTE) AS QTE_MOYENNE FROM COMPREND GROUP BY NUMCONS;
24 SELECT VILLESERVEUR,COUNT(*) AS NOMBRE_SERVEUR FROM SERVEUR GROUP BY VILLESERVEUR;
25 SELECT VILLESERVEUR,COUNT(*) AS NOMBRE_SERVEUR FROM SERVEUR GROUP BY VILLESERVEUR HAVING COUNT(
       VILLESERVEUR) > 1;
26 SELECT NUMFACTURE,COUNT(*) AS NOMBRES_CONSOMMATIONS FROM COMPREND GROUP BY NUMFACTURE;
27 SELECT NUMFACTURE,SUM(QTE) AS NOMBRES_CONSOMMATIONS FROM COMPREND GROUP BY NUMFACTURE;
28 SELECT NUMCONS,COUNT(*) AS NB_FACTURE FROM COMPREND GROUP BY NUMCONS;
29 SELECT NUMCONS,COUNT(*) AS NB_FACTURE FROM COMPREND GROUP BY NUMCONS HAVING COUNT(*) > 2;
```

```
30 SELECT * FROM SERVEUR ORDER BY VILLESERVEUR,NOMSERVEUR;
31 SELECT * FROM SERVEUR ORDER BY VILLESERVEUR DESC,NOMSERVEUR;
32 SELECT NUMFACTURE,NUMTABLE,NOMSERVEUR FROM (FACTURE JOIN SERVEUR USING(NUMSERVEUR));
33 SELECT NUMFACTURE,NOMSERVEUR FROM (FACTURE JOIN SERVEUR USING(NUMSERVEUR)) WHERE NUMTABLE = 5;
34 SELECT NUMFACTURE,NOMTABLE,NOMSERVEUR FROM ((FACTURE JOIN LESTABLES USING(NUMTABLE)) JOIN SERVEUR USING
       (NUMSERVEUR));
35 SELECT DISTINCT NOMSERVEUR,NOMTABLE FROM ((FACTURE JOIN LESTABLES USING(NUMTABLE)) JOIN SERVEUR USING(
       NUMSERVEUR)) ORDER BY NOMSERVEUR;
36 SELECT NUMCONS,LIBCONS,PRIXCONS,QTE FROM (COMPREND JOIN CONSOMMATION USING(NUMCONS)) WHERE NUMFACTURE =
       1203;
37 SELECT NUMCONS,LIBCONS,PRIXCONS,QTE FROM (((FACTURE JOIN LESTABLES USING(NUMTABLE)) JOIN COMPREND USING
       (NUMFACTURE)) JOIN CONSOMMATION USING (NUMCONS)) WHERE DATEFACTURE = '01/02/2010' AND NUMTABLE = 5;
38 SELECT NOMTABLE,NUMFACTURE FROM (LESTABLES LEFT JOIN FACTURE USING(NUMTABLE));
39 SELECT NOMTABLE,NUMFACTURE FROM (FACTURE RIGHT JOIN LESTABLES USING(NUMTABLE));
40 SELECT NUMTABLE,NOMTABLE FROM (LESTABLES LEFT JOIN FACTURE USING(NUMTABLE)) WHERE NUMFACTURE IS NULL;
41 SELECT NUMCONS,LIBCONS FROM (((SELECT NUMSERVEUR,NUMFACTURE FROM FACTURE JOIN SERVEUR USING(NUMSERVEUR)
       ) JOIN COMPREND USING(NUMFACTURE)) LEFT JOIN CONSOMMATION USING(NUMCONS)) WHERE NUMSERVEUR = 52;
42 SELECT DISTINCT NUMCONS,LIBCONS FROM CONSOMMATION MINUS (SELECT NUMCONS,LIBCONS FROM (COMPREND JOIN
       CONSOMMATION USING(NUMCONS)));
43 SELECT NUMFACTURE,DATEFACTURE,SUM(QTE) AS NB_CONS FROM (COMPREND JOIN FACTURE USING(NUMFACTURE)) GROUP
       BY NUMFACTURE,DATEFACTURE;
44 SELECT NUMFACTURE,SUM(PRIXCONS*QTE) AS PRIX_TOTAL FROM (COMPREND JOIN CONSOMMATION USING(NUMCONS))
       GROUP BY NUMFACTURE;
45 SELECT DATEFACTURE,SUM(QTE) AS NB_CONS FROM (COMPREND JOIN FACTURE USING(NUMFACTURE)) GROUP BY
       DATEFACTURE;
46 SELECT DATEFACTURE,SUM(PRIXCONS*QTE) AS CA FROM ((COMPREND JOIN CONSOMMATION USING(NUMCONS)) LEFT JOIN
       FACTURE USING(NUMFACTURE)) GROUP BY DATEFACTURE;
47 SELECT NOMSERVEUR,COUNT(*) AS NB_FACTURES FROM (FACTURE LEFT JOIN SERVEUR USING(NUMSERVEUR)) GROUP BY
       NOMSERVEUR;
48 SELECT NOMSERVEUR,COUNT(*) AS NB_FACTURES FROM ((COMPREND JOIN FACTURE USING(NUMFACTURE)) RIGHT JOIN
       SERVEUR USING(NUMSERVEUR)) GROUP BY NOMSERVEUR;
49 SELECT NOMSERVEUR,SUM(PRIXCONS*QTE) AS CA FROM (((COMPREND JOIN FACTURE USING(NUMFACTURE)) JOIN SERVEUR
       USING(NUMSERVEUR)) JOIN CONSOMMATION USING(NUMCONS)) GROUP BY NOMSERVEUR;
50 SELECT NOMTABLE,COUNT(*) AS NB_FACTURES FROM (FACTURE JOIN LESTABLES USING(NUMTABLE)) GROUP BY NOMTABLE
       ;
51 SELECT LIBCONS,COUNT(*) AS NB_FACTURES FROM ((COMPREND JOIN FACTURE USING(NUMFACTURE)) JOIN
       CONSOMMATION USING(NUMCONS)) GROUP BY LIBCONS;
52 SELECT NOMTABLE,SUM(PRIXCONS*QTE) FROM (((COMPREND JOIN FACTURE USING(NUMFACTURE)) JOIN LESTABLES USING
       (NUMTABLE)) JOIN CONSOMMATION USING(NUMCONS)) GROUP BY NOMTABLE;
```

Voici les résultats de ces commandes :

Requête 2	Requête 19	Requête 36
Requête 3	Requête 20	Requête 37
Requête 4	Requête 21	Requête 38
Requête 5	Requête 22	Requête 39
Requête 6	Requête 23	Requête 40
Requête 7	Requête 24	Requête 41
Requête 8	Requête 25	Requête 42
Requête 9	Requête 26	Requête 43
Requête 10	Requête 27	Requête 44
Requête 11	Requête 28	Requête 45
Requête 12	Requête 29	Requête 46
Requête 13	Requête 30	Requête 47
Requête 14	Requête 31	Requête 48
Requête 15	Requête 32	Requête 49
Requête 16	Requête 33	Requête 50
Requête 17	Requête 34	Requête 51
Requête 18	Requête 35	Requête 52

TABLE 5.1 – Images requêtes TP5

5.1.2 Généalogie royale

1) Fichier genealogie.sql

```
CREATE TABLE genealogie (
   numPer                NUMERIC,
   Nom                   varchar2(35) NOT NULL,
   DateNaissance         date NOT NULL,
   Pere                  NUMERIC DEFAULT NULL,
   Mere                  NUMERIC DEFAULT NULL,
  PRIMARY KEY (numPer)
);
```

2) Soit la table suivante :

numPer	Nom	DateNaissance	Pere	Mere
1	George VI	1895-12-14		
2	Elizabeth Bowes-Lyon	1900-08-04		
3	Elizabeth II	1926-04-21	1	2
4	Margaret du Royaume-Uni	1921-06-10	1	2
5	Philip Mountbatten	1921-06-10		
6	Prince Charles	1948-11-14	5	3
7	Princesse Anne	1950-08-15	5	3
8	Prince Andrew	1960-02-19	5	3
9	Prince Edward	1964-03-10	5	3
10	Diana Spencer	1961-07-01		
11	Prince William	1982-06-21	6	10
12	Prince Henry	1984-09-15	6	10

TABLE 5.2 – Généalogie

Voici quelques requêtes

1. Affiche le nom et la date de naissance des enfants d'Élizabeth II. Image

```
SELECT Nom, DateNaissance FROM GENEALOGIE WHERE Mere = 3;
```

2. Affiche la mère du Prince William. Image

```
SELECT G2.NOM, G2.DATENAISSANCE FROM GENEALOGIE G1, GENEALOGIE G2 WHERE G1.NUMPER = 11 AND G2.
    NUMPER = G1.MERE;
```

3. Affiche les parents d'Élizabeth II. Image

```
SELECT G2.NOM, G2.DATENAISSANCE FROM GENEALOGIE G1, GENEALOGIE G2 WHERE (G1.NUMPER = 3 AND G2.
    NUMPER = G1.MERE) OR (G1.NUMPER = 3 AND G2.NUMPER = G1.PERE);
```

4. Affiche les frères et soeurs du Prince Charles. Image

```
SELECT G2.NOM, G2.DATENAISSANCE FROM GENEALOGIE G1, GENEALOGIE G2 WHERE (G1.NUMPER = 6 AND G2.MERE
    = G1.MERE AND G2.PERE = G1.PERE AND G2.NUMPER != 6);
```

5. Affiche le nom du père, le nom de la mère et le nom de l'individu. (affiche `NULL` si `Pere` où `Mere` = `NULL`) Image

```
SELECT G1.NOM, G2.NOM AS NOM_PERE, G3.NOM AS NOM_MERE FROM GENEALOGIE G1, GENEALOGIE G2, GENEALOGIE
    G3 WHERE (G2.NUMPER = G1.PERE AND G3.NUMPER = G1.MERE) UNION (SELECT NOM, NULL, NULL FROM
    GENEALOGIE WHERE PERE IS NULL OR MERE IS NULL);
```

G1 le nom de la personne, G2 le nom de la mère et G3 le nom du père. La requête à gauche de `UNION` affiche toutes les personnes ainsi que les noms de leurs parents. La requête à droite de `UNION` affiche toutes les personnes qui n'ont pas de parents, et l'union des deux sous-requêtes affiche toutes les personnes ainsi que leurs parents (ou `NULL` si pas de parents).

6. Affiche le nom de l'individu et le nombre d'enfents de cet individu. On affiche seulement si l'individu possède au moins un enfant. Image

```
1 SELECT G1.NOM,COUNT(*) AS NB_ENFANTS FROM GENEALOGIE G1,GENEALOGIE G2 WHERE G1.NUMPER = G2.PERE
      OR G1.NUMPER = G2.MERE GROUP BY G1.NOM;
```

Chapitre 6

TP6

6.1 SQL : Requêtes avancées

Liste des requêtes de l'annexe B Appendice B

Voici les fichiers utilisés :
- ecole_tp6.sql
- requetes_tp6.sql

Voici les contenus de nos tables :
- La table `ELEVES` Image
- La table `PROFESSEURS` Image
- La table `COURS` Image
- La table `CHARGE` Image
- La table `RESULTATS` Image
- La table `ACTIVITES` Image
- La table `ACTIVITES_PRATIQUEES` Image

Voici la liste des commandes :

```
1  SELECT NOM,PRENOM,DATE_NAISSANCE FROM ELEVES;
2  SELECT * FROM ACTIVITES;
3  SELECT SPECIALITE FROM PROFESSEURS;
4  SELECT NOM,PRENOM FROM ELEVES WHERE POIDS < 45 AND ANNEE BETWEEN 1 AND 2;
5  SELECT NOM FROM ELEVES WHERE POIDS BETWEEN 60 AND 80;
6  SELECT NOM FROM PROFESSEURS WHERE SPECIALITE = 'poésie' OR SPECIALITE = 'sql';
7  SELECT NOM FROM ELEVES WHERE NOM LIKE 'L%';
8  SELECT NOM FROM PROFESSEURS WHERE SPECIALITE IS NULL;
9  SELECT NOM,PRENOM FROM ELEVES WHERE POIDS < 45 AND ANNEE = 1;
10 SELECT NOM,DECODE(SPECIALITE,NULL,'****',SPECIALITE) AS SPECIALITE FROM PROFESSEURS;
11 /* 11 */
12 SELECT E.NOM,E.PRENOM FROM (ELEVES E JOIN ACTIVITES_PRATIQUEES A USING(Num_eleve)) WHERE (NIVEAU = 1
       AND A.NOM = 'Surf');
13 SELECT E.NOM,E.PRENOM FROM ((SELECT * FROM ACTIVITES_PRATIQUEES WHERE NIVEAU = 1 AND NOM = 'Surf') JOIN
       ELEVES E USING(NUM_ELEVE));
14 SELECT E.NOM,E.PRENOM FROM ((SELECT * FROM ACTIVITES_PRATIQUEES WHERE NIVEAU IN (1) AND NOM IN ('Surf')
       ) JOIN ELEVES E USING(NUM_ELEVE));
15 SELECT E.NOM,E.PRENOM FROM (((SELECT * FROM ACTIVITES_PRATIQUEES) MINUS (SELECT * FROM
       ACTIVITES_PRATIQUEES WHERE NIVEAU != 1 OR NOM != 'Surf')) JOIN ELEVES E USING(NUM_ELEVE));
16 SELECT E.NOM,E.PRENOM FROM (((SELECT * FROM ACTIVITES_PRATIQUEES) INTERSECT (SELECT * FROM
       ACTIVITES_PRATIQUEES WHERE NIVEAU IN (1) AND NOM IN ('Surf'))) JOIN ELEVES E USING(NUM_ELEVE));
17 /* 12 */
18 SELECT E.NOM FROM ((ACTIVITES_PRATIQUEES AP JOIN ACTIVITES A USING(Niveau,Nom)) JOIN ELEVES E USING(
       Num_eleve)) WHERE EQUIPE = 'Amc_Indus';
19 SELECT E.NOM FROM ((SELECT * FROM ACTIVITES A WHERE EQUIPE IN ('Amc_Indus')) JOIN ACTIVITES_PRATIQUEES
       AP USING(NOM,NIVEAU)) JOIN ELEVES E USING(NUM_ELEVE);
20 /* 13 */
21 SELECT P1.NOM,P2.NOM FROM PROFESSEURS P1, PROFESSEURS P2 WHERE (P1.SPECIALITE = P2.SPECIALITE AND P1.
       NUM_PROF != P2.NUM_PROF);
```

```
/* 14 */
SELECT NOM,SALAIRE_ACTUEL,SALAIRE_ACTUEL-SALAIRE_BASE AS AUGMENTATION FROM PROFESSEURS WHERE SPECIALITE = 'sql';
SELECT NOM,SALAIRE_ACTUEL,SALAIRE_ACTUEL-SALAIRE_BASE AS AUGMENTATION FROM PROFESSEURS WHERE SPECIALITE IN ('sql');
/* 15 */
SELECT NOM FROM PROFESSEURS WHERE SALAIRE_ACTUEL-SALAIRE_BASE > 0.25*SALAIRE_BASE;
SELECT NOM FROM PROFESSEURS WHERE SALAIRE_ACTUEL-SALAIRE_BASE != 0 AND SALAIRE_ACTUEL-SALAIRE_BASE BETWEEN 0 AND 0.25*SALAIRE_BASE;
/* 16 */
SELECT 4*POINTS AS POINTS_SUR_100 FROM (RESULTATS JOIN ELEVES USING(NUM_ELEVE)) WHERE NUM_ELEVE = 5;
SELECT 4*R.POINTS AS POINTS_SUR_100 FROM ((SELECT * FROM RESULTATS) MINUS (SELECT * FROM RESULTATS WHERE NUM_ELEVE != 5)) R;
/* 17 */
SELECT AVG(POIDS) AS POIDS_MOYEN_AN1 FROM ELEVES WHERE ANNEE = 1;
SELECT AVG(POIDS) AS POIDS_MOYEN_AN1 FROM ((SELECT * FROM ELEVES) MINUS (SELECT * FROM ELEVES WHERE ANNEE != 1));
/* 18 */
SELECT SUM(POINTS) AS POINT_TOTAL FROM RESULTATS WHERE NUM_ELEVE = 3;
SELECT SUM(POINTS) AS POINT_TOTAL FROM ((SELECT * FROM RESULTATS) INTERSECT (SELECT * FROM RESULTATS WHERE NUM_ELEVE = 3));
/* 19 */
SELECT MIN(POINTS) AS MINIMUN, MAX(POINTS) AS MAXIMUM FROM RESULTATS WHERE NUM_ELEVE = 1;
SELECT MIN(POINTS) AS MINIMUN, MAX(POINTS) AS MAXIMUM FROM RESULTATS WHERE NUM_ELEVE IN (1);
/* 20 */
SELECT COUNT(*) AS NB_ELEVES_AN2 FROM ELEVES GROUP BY ANNEE HAVING ANNEE = 2;
SELECT COUNT(*) AS NB_ELEVES_AN2 FROM ELEVES WHERE ANNEE = 2;
/* 21 */
SELECT AVG(SALAIRE_ACTUEL-SALAIRE_BASE) AS MOYENNE_AUGMENTATION FROM PROFESSEURS WHERE SPECIALITE = 'sql';
SELECT AVG(SALAIRE_ACTUEL-SALAIRE_BASE) AS MOYENNE_AUGMENTATION FROM PROFESSEURS WHERE SPECIALITE IN ('sql');
/* 22 */
SELECT EXTRACT(YEAR FROM Der_prom) AS ANNEE_DER_PROM FROM PROFESSEURS WHERE NUM_PROF = 8;
/* 23 */
SELECT Date_entree,Der_prom,EXTRACT(YEAR FROM Der_prom)-EXTRACT(YEAR FROM Date_entree) AS ANNEE_PASSEE FROM PROFESSEURS;
/* 24 */
SELECT AVG(EXTRACT(YEAR FROM SYSDATE)-EXTRACT(YEAR FROM date_naissance)) AS AGE_MOYEN FROM ELEVES;
/* 25 */
SELECT NOM FROM PROFESSEURS WHERE ADD_MONTHS(Date_entree,50) < Der_prom;
/* 26 */
SELECT NOM FROM ELEVES WHERE EXTRACT(YEAR FROM ADD_MONTHS(SYSDATE,4)) - EXTRACT(YEAR FROM date_naissance) > 24;
/* 27 */
SELECT * FROM ELEVES ORDER BY ANNEE,NOM;
/* 28 */
SELECT 4*POINTS AS POINTS_SUR_100 FROM RESULTATS WHERE NUM_ELEVE = 5 ORDER BY 4*POINTS DESC;
SELECT 4*POINTS AS POINTS_SUR_100 FROM ((SELECT * FROM RESULTATS) MINUS (SELECT * FROM RESULTATS WHERE NUM_ELEVE != 5)) ORDER BY 4*POINTS DESC;
/* 29 */
SELECT NOM,AVG(POINTS) AS MOYENNE FROM (RESULTATS JOIN ELEVES USING(NUM_ELEVE)) WHERE ANNEE = 1 GROUP BY NOM;
/* 30 */
SELECT NUM_ELEVE,AVG(POINTS) AS MOYENNE FROM (RESULTATS JOIN ELEVES USING(NUM_ELEVE)) WHERE ANNEE = 1 GROUP BY NUM_ELEVE HAVING SUM(POINTS) > 40;
/* 31 */
SELECT MAX(POINT_TOTAL) AS MAXIMUM FROM (SELECT SUM(POINTS) AS POINT_TOTAL FROM (RESULTATS JOIN ELEVES USING(NUM_ELEVE)) GROUP BY NUM_ELEVE);
/* 32 */
SELECT E.NOM FROM ((ACTIVITES_PRATIQUEES AP JOIN ACTIVITES USING(NIVEAU,NOM)) JOIN ELEVES E USING(NUM_ELEVE)) WHERE EQUIPE = 'Amc Indus';
/* 33 */
SELECT AVG(POINTS) AS MOYENNE FROM (RESULTATS JOIN ELEVES USING(NUM_ELEVE)) WHERE ANNEE = 1 GROUP BY NUM_ELEVE HAVING AVG(POINTS) > (SELECT AVG(POINTS) AS MOYENNE FROM (RESULTATS JOIN ELEVES USING(NUM_ELEVE)) GROUP BY ANNEE HAVING ANNEE = 1);
/* 34 */
SELECT E1.NOM, E1.POIDS FROM ELEVES E1 WHERE ANNEE = 1 GROUP BY NOM, POIDS HAVING E1.POIDS > (SELECT MAX(POIDS) FROM ELEVES E2 WHERE ANNEE = 2);
SELECT E1.NOM, E1.POIDS FROM ELEVES E1 WHERE ANNEE = 1 AND NOT EXISTS (SELECT E2.NOM, E2.POIDS FROM ELEVES E2 WHERE E1.POIDS < E2.POIDS) GROUP BY NOM, POIDS;
```

```
74 /* 35 */
75 SELECT E1.NOM, E1.POIDS FROM ELEVES E1 WHERE ANNEE = 1 GROUP BY NOM, POIDS HAVING E1.POIDS > (SELECT MIN(
       POIDS) FROM ELEVES E2 WHERE ANNEE = 2);
76 SELECT E1.NOM, E1.POIDS FROM ELEVES E1 WHERE ANNEE = 1 AND E1.POIDS > ANY (SELECT E2.POIDS FROM ELEVES
       E2 WHERE ANNEE = 2);
77 /* 36 */
78 SELECT NOM, POIDS, ANNEE FROM ELEVES E1 WHERE E1.POIDS > (SELECT AVG(POIDS) FROM ELEVES E2 WHERE E2.ANNEE
       = E1.ANNEE);
79 /* 37 */
80 SELECT NOM FROM ((SELECT * FROM (PROFESSEURS JOIN CHARGE USING(NUM_PROF))) MINUS (SELECT * FROM (
       PROFESSEURS JOIN CHARGE USING(NUM_PROF)) WHERE NUM_COURS != 1));
81 /* 38 */
82 SELECT DISTINCT NOM FROM ELEVES NATURAL JOIN RESULTATS NATURAL JOIN (SELECT NUM_ELEVE FROM
       ACTIVITES_PRATIQUEES AP WHERE AP.NOM = 'Tennis') NATURAL JOIN (SELECT NUM_ELEVE FROM RESULTATS
       GROUP BY NUM_ELEVE HAVING AVG(POINTS) > 0.6*20) WHERE ANNEE = 1;
83 /* 39 */
84 SELECT DISTINCT NUM_PROF,NOM FROM PROFESSEURS P WHERE
85       (SELECT COUNT(*) FROM (CHARGE C1 JOIN COURS C2 USING(NUM_COURS)) WHERE NUM_PROF = P.NUM_PROF AND
             ANNEE = 2)
86       =
87       (SELECT COUNT(*) FROM COURS WHERE ANNEE = 2)
88 ;
89 /* 40 */
90 SELECT NUM_ELEVE,NOM FROM ELEVES E WHERE
91       (SELECT COUNT(DISTINCT NOM) FROM (ACTIVITES_PRATIQUEES AP JOIN ACTIVITES A USING(NIVEAU,NOM)) WHERE
             AP.NUM_ELEVE = E.NUM_ELEVE)
92       =
93       (SELECT COUNT(DISTINCT NOM) FROM ACTIVITES)
94 ;
```

Voici les résultats de ces commandes :

Requête 1	Requête 11	Requête 21	Requête 31
Requête 2	Requête 12	Requête 22	Requête 32
Requête 3	Requête 13	Requête 23	Requête 33
Requête 4	Requête 14	Requête 24	Requête 34
Requête 5	Requête 15	Requête 25	Requête 35
Requête 6	Requête 16	Requête 26	Requête 36
Requête 7	Requête 17	Requête 27	Requête 37
Requête 8	Requête 18	Requête 28	Requête 38
Requête 9	Requête 19	Requête 29	Requête 39
Requête 10	Requête 20	Requête 30	Requête 40

TABLE 6.1 – Images requêtes TP6

Note : Certaines requêtes possèdent plusieurs implémentations possibles, si c'est le cas le screenshot de la requête en question et le screenshot de la première implémentation de cette requête (les screenshots affichent le même résultat). Voici quelques explications sur les requêtes difficiles :

34) On cherche les élèves de première année qui sont plus lourds que n'importe quels élèves de deuxièmes années. C'est-à-dire un élève de première année qui est plus lourd que le plus lourd des élèves de deuxièmes années. Or aucun élève de première année est plus lourd que le plus lourd de deuxièmes années. S'il est plus lourd que le plus lourd des élèves de deuxièmes années, alors il est plus lourd que n'importe quels élèves de deuxièmes années.

35) On cherche les élèves de première année qui sont plus lourds qu'un élève quelconque de deuxièmes années. C'est-à-dire un élève de première année qui est plus lourd que le moins lourd des élèves de deuxièmes années. S'il est plus lourd que le moins lourd des élèves de deuxièmes années, alors il est plus lourd qu'un élève quelconque de deuxièmes années.

36) On regarde si l'élève à un poids supérieur aux moyennes des poids de leurs années respectif. On teste donc la moyenne des poids avec une sous-requête qui renvoie la moyenne des poids des élèves de la même année que l'élève concerné.

37) On choisit tous les professeurs et on enlève tous les professeurs qui donnent le cours numéro 1 (même s'ils ont d'autres cours que le cours numéro 1).

38) On choisit que les élèves qui pratiquent le tennis, puis on choisit parmi les élèves restants, les élèves qui ont une note de plus de 60%, et on ne garde que les élèves qui sont en première année.

39) Il s'agit d'une division. La première sous-requête affiche le nombre de cours de deuxièmes années que chaque professeur donne, la deuxième sous-requête affiche le nombre total de cours de deuxièmes années. On affiche donc tous les professeurs qui donnent le nombre de cours de deuxièmes années égale au nombre total de cours de deuxièmes années.

40) Il s'agit d'une division. La première sous-requête affiche le nombre d'activités que pratiquent chaque élève, la deuxième sous-requête affiche le nombre total d'activités. On affiche donc tous les élèves qui pratiquent le nombre d'activités égale au nombre d'activité totale.

Chapitre 7

TP7

7.1 SQL : Vues et Arbres

1) Ficher famille.sql

NUMERO	NOM	PRENOM	DATENAISSANCE	SEXE	PERE	MERE
99	LEBON	NICOLAS	01/01/1895	M		
88	HONNEUR	CLEMENCE	01/01/1900	F		
1	LEBON	MICHEL	08/04/1920	M	99	88
15	LEBON	GABRIEL	08/04/1936	M	99	88
98	CLEMENT	JEAN-BAPTISTE	01/01/1890	M		
87	GABRIEL	EVE	01/01/1892	F		
2	CLEMENT	EVE	11/13/1928	F	98	87
22	CLEMENT	JEAN-BAPTISTE	11/13/1910	F	98	
3	LEBON	NICOLAS	09/17/1958	M	1	2
33	LEBON	ROSE	06/16/1951	F	1	2
34	CLEMENT	RAOUL	01/01/1941	M	22	
55	CLEMENT	MARIE	08/13/1978	F	33	34
56	MEDECIN	LINA	02/22/2002	F	55	
77	PARIS	LOUIS	03/20/1924	M		
78	GATEAU	EVELYNE	03/20/1936	F		
4	PARIS	INES	11/22/1969	F	77	78
76	PARIS	AMELIA	10/20/1958	F	77	78
75	AMMAR	SERGES	10/20/1987	M		76
5	LEBON	CLEMENCE	06/19/2001	F	3	4
6	LEBON	ADAM	06/19/2001	M	3	4
7	LEBON	FRANCOIS	02/22/1954	M	1	2
9	LEBON	FRANCOISE	09/01/1963	F	15	
8	LEBON	MICHEL	09/05/1987	M	7	9
10	LEBON	AIME	05/24/1993	M	7	9
11	LEBON	ALEXANDRE	07/16/1994	M	7	9

Table 7.1 – Personnes

2) Le schémas E/A de la table : Image

Remarque : La commande `&num` ne fonctionnant pas sur `https://apex.oracle.com`, on choisit donc dans la requête une personne.

3) Créer une vue permettant d'afficher les ancêtres d'une personne avec la commande `CONNECT BY`. Le numéro de la personne doit être demandé au moment où la requête se lance (`&num`). On choisit la personne numéro 7. Image

```
DROP VIEW TEST;
CREATE VIEW TEST(NUMERO,NOM,PRENOM,PERE,MERE) AS (
    SELECT NUMERO,NOM,PRENOM,PERE,MERE FROM PERSONNES WHERE NUMERO != 7
        START WITH NUMERO = 7
        CONNECT BY NUMERO = PRIOR MERE OR NUMERO = PRIOR PERE)
;
SELECT * FROM TEST;
```

4) Affichez le père, le grand-père et l'arrière-grand-père d'une personne `&num` en n'utilisant pas `CONNECT BY` mais en utilisant plusieurs vues intermédiaires. On choisit la personne numéro 10. Image

```
DROP VIEW PERE;
DROP VIEW GRAND_PERE;
DROP VIEW ARRIERE_GRAND_PERE;
CREATE VIEW PERE(PERE) AS (SELECT PERE FROM PERSONNES P WHERE P.NUMERO = 10);
CREATE VIEW GRAND_PERE(PERE) AS (SELECT PERE FROM PERSONNES P WHERE P.NUMERO IN (SELECT PERE FROM PERE)
    );
CREATE VIEW ARRIERE_GRAND_PERE(PERE) AS (SELECT PERE FROM PERSONNES P WHERE P.NUMERO IN (SELECT PERE
    FROM GRAND_PERE));
SELECT * FROM PERE;
SELECT * FROM GRAND_PERE;
SELECT * FROM ARRIERE_GRAND_PERE;
SELECT * FROM PERSONNES WHERE NUMERO IN (SELECT PERE FROM PERE) OR NUMERO IN (SELECT PERE FROM
    GRAND_PERE) OR NUMERO IN (SELECT PERE FROM ARRIERE_GRAND_PERE);
```

On a utilisé 3 vues intermédiaires :

- La vue `PERE` qui renvoie le numéro du père de la personne 10. Image
- La vue `GRAND_PERE` qui renvoie le numéro du grand-père de la personne 10. Image
- La vue `ARRIERE_GRAND_PERE` qui renvoie le numéro de l'arrière-grand-père de la personne 10. Image

Puis on affiche les personnes qui ont leurs numéros qui sont dans les vues `PERE`, `GRAND_PERE` et `ARRIERE_GRAND_PERE`.

5) Affiche tous les descendants d'une personne. On affiche le numéro, le nom, le prénom ainsi que l'arborescence. On choisit la personne numéro 98. Image

```
SELECT NUMERO,NOM,PRENOM FROM PERSONNES WHERE NUMERO != 98
START WITH NUMERO = 98
CONNECT BY PRIOR NUMERO = MERE OR PRIOR NUMERO = PERE;
```

6) Affiche les frères et soeurs d'une personne. On choisit la personne numéro 8. Image

```
SELECT P2.NUMERO, P2.NOM, P2.PRENOM, P1.MERE, P1.PERE FROM PERSONNES P1,PERSONNES P2 WHERE P1.NUMERO = 8
    AND P1.MERE = P2.MERE AND P1.PERE = P2.PERE AND P1.NUMERO != P2.NUMERO;
```

Il suffit de regarder quels sont les personnes qui ont le même père et la même mère que la personne 8.

7) Affiche seulement les soeurs d'une personne. On choisit la personne numéro 7. Image

```
SELECT P2.NUMERO, P2.NOM, P2.PRENOM, P1.MERE, P1.PERE FROM PERSONNES P1,PERSONNES P2 WHERE P1.NUMERO = 7
    AND P1.MERE = P2.MERE AND P1.PERE = P2.PERE AND P1.NUMERO != P2.NUMERO AND P2.SEXE = 'F';
```

Il suffit de regarder quels sont les personnes qui ont le même père et la même mère que la personne 8 et qui ont `SEXE = 'F'`.

8) Affiche tous les enfants des femmes de plus de 40 ans. On commence par créer une vue contenant que les femmes de 40 ans, puis on affiche tous les personnes qui ont pour mère une valeur contenant dans la vue. Image

```
CREATE VIEW TEST(NUMERO,PERE,MERE) AS (SELECT NUMERO,PERE,MERE FROM PERSONNES WHERE SEXE = 'F' AND
    EXTRACT(YEAR FROM SYSDATE)-EXTRACT(YEAR FROM DATENAISSANCE) > 40);
SELECT * FROM PERSONNES P WHERE P.MERE IN (SELECT NUMERO FROM TEST);
```

On a crée une vue `TEST` qui renvoie toutes les femmes qui ont plus de 40 ans. Puis on affiche les personnes qui ont leurs mères qui sont dans la vue `TEST`. Image

9) Affichez les cousins et cousines d'une personne. On choisit la personne numéro 8. Image

```
1 DROP VIEW PARENTS;
2 DROP VIEW ONCLES_TANTES;
3 CREATE VIEW PARENTS(NUMERO,PERE,MERE) AS (SELECT P2.NUMERO,P2.PERE,P2.MERE FROM PERSONNES P1,PERSONNES
      P2 WHERE (P1.NUMERO = 8 AND P1.NUMERO != P2.NUMERO) AND (P1.PERE = P2.NUMERO OR P1.MERE = P2.NUMERO
      ));
4 CREATE VIEW ONCLES_TANTES(NUMERO) AS (SELECT NUMERO FROM PERSONNES P WHERE (P.MERE IN (SELECT MERE FROM
       PARENTS) OR P.PERE IN (SELECT PERE FROM PARENTS)) AND P.NUMERO NOT IN (SELECT NUMERO FROM PARENTS)
      );
5 SELECT * FROM PARENTS;
6 SELECT * FROM ONCLES_TANTES;
7 SELECT * FROM PERSONNES P WHERE P.PERE IN (SELECT NUMERO FROM ONCLES_TANTES) OR P.MERE IN (SELECT
      NUMERO FROM ONCLES_TANTES);
```

On a crée 2 vues intermédiaires :

- La vue `PARENTS` qui renvoie le père et la mère de la personne 8. Image
- La vue `ONCLES_TANTES` qui renvoie les oncles et les tantes de la personne 8.Image

Puis on affiche les personnes qui ont leur mère ou père qui sont dans la vue `ONCLES_TANTES`.

10) Affichez les cousins issus de germain d'une personne.

11) Affichez les petits enfants de la personne la plus âgée ayant des petits enfants. Image

```
1 DROP VIEW MAX_AGE;
2 DROP VIEW ENFANTS;
3 CREATE VIEW MAX_AGE(NUMERO) AS (SELECT NUMERO FROM PERSONNES WHERE DATENAISSANCE = (SELECT MIN(
      DATENAISSANCE) FROM PERSONNES));
4 CREATE VIEW ENFANTS(NUMERO) AS (SELECT NUMERO FROM PERSONNES P WHERE P.PERE IN (SELECT NUMERO FROM
      MAX_AGE) OR P.MERE IN (SELECT NUMERO FROM MAX_AGE));
5 SELECT * FROM MAX_AGE;
6 SELECT * FROM ENFANTS;
7 SELECT NUMERO FROM PERSONNES P WHERE P.PERE IN (SELECT NUMERO FROM ENFANTS) OR P.MERE IN (SELECT NUMERO
       FROM ENFANTS);
```

On a créer 2 vues intermédiaires :

- La vue `MAX_AGE` qui renvoie le numéro de la personne la plus âgée dans la table. Image
- La vue `ENFANTS` qui renvoie les enfants de la personne la plus âgée. Image

Puis on affiche tous les personnes qui ont leur père ou mère dans la vue `ENFANTS` (les petits-enfants).

12) Pour chaque personne, affichez les noms, prénom et âge de son descendant le plus jeune.

Annexe A

Requêtes TP5

1. Liste du contenu de chaque table de la base.
2. Nombre de places de la table numéro 4 (Nbplace).
3. Liste des consommations dont le prix unitaire est supérieur à 1 euro (Numcons, Libcons, Prixcons).
4. Liste des serveurs de Belfort et de Delle (Numserveur, Nomserveur, Villeserveur).
5. Liste des factures du 2 février servies par le serveur 52 (Numfacture, Numtable).
6. Liste des consommations de la facture 1203 (Numcons, Qte).
7. Liste des consommations des factures 1200 et 1201 (sans lignes en double) (Numcons).
8. Liste des serveurs qui sont nés en 1976 (Nomserveur, Dateserveur).
9. Liste des consommations de type bière (Numcons, Libcons, Prixcons).
10. Liste des tables servies après le 1er février.
11. Liste des serveurs dont le nom contient i en deuxième position (Nomserveur).
12. Liste des serveurs dont le nom commence par un P (Nomserveur).
13. Liste des serveurs par ville (Nomserveur, Villeserveur).
14. Liste des consommations classées par ordre alphabétique sur le libellé (Libcons, Numcons, Prixcons).
15. Liste des villes où habitent les serveurs (sans lignes en double) (Villeserveur).
16. Le nombre de tables du restaurant.
17. Le nombre de places disponibles sur l'ensemble des tables.
18. Nombre de factures établies par chaque serveur (Numserveur, Nbfacture).
19. Nombre de factures établies chaque jour (Datefacture, Nbfacture).
20. Liste des serveurs qui ont établi plus de 3 factures (Numserveur, Nbfacture).
21. Prix moyen des consommations (Prixmoyen).
22. Prix moyen du café (Prixmoyen).
23. Quantité moyenne consommé pour chaque consommation (Numcons, Qtemoyenne).
24. Nombre de serveur par ville (Villeserveur, Nbserveur).
25. Liste des villes dans lesquelles habitent plus d'un serveur (Villeserveur, Nbserveur).
26. Nombre de types de consommations par factures (Numfacture, Nbcons).
27. Nombre total de consommations (en comptant la quantité) par facture (Numfacture, Qtecons).
28. Nombre de factures par consommation (Numcons, Nbfacture).
29. Consommations qui interviennent dans plus de 2 factures (Numcons, Nbfactures).
30. Liste des serveurs, triés par nom de ville croissante, puis le nom de serveur croissant.
31. Liste des serveurs triés par nom de ville décroissante, puis nom de serveur croissant.
32. Liste des factures avec leur numéro de table et le nom du serveur (Numfacture, Numtable, Nomserveur).
33. Liste des factures de la table 5 avec le nom du serveur (Numfacture, Nomserveur).

34. Liste des factures avec leur nom de table et le nom du serveur (Numfacture, Nomtable, Nomserveur).
35. Liste des serveurs et des tables qu'ils ont servies ordonnés selon le nom du serveur (pas de ligne double) (Nomserveur, Nomtable).
36. Liste des consommations de la facture 1203 avec leur nom, leur prix et leur quantité (Numcons, Libcons, Prixcons, Qte).
37. Liste des consommations du premier février de la table 5 avec leur nom, leur prix et leur quantité (Numcons, Libcons, Prixcons, Qte).
38. Liste des tables et des numéros de factures qui leur sont associées. Attention, on veut voir toutes les tables même si elles n'ont pas de factures. La table de départ (celle du `FROM`) sera `LESTABLES` (Nomtable, Numfacture).
39. Même question que précédemment, mais avec `FACTURE` comme table de départ.
40. Liste des tables qui n'ont eu aucune facture (Numtable, Nomtable).
41. Liste des consommations qui ont déjà été servies par le serveur 52 (Numcons, Libcons).
42. Liste des consommations qui n'ont jamais été servis (Numcons, Libcons).
43. La liste des factures avec leur date et leur nombre de consommations (prendre en compte la quantité) (Numfacture, Datefacture, Nbcons).
44. La liste des factures et le montant de leur addition (Numfacture, Prixfacture).
45. Nombre de consommations servies par jour (Datefacture, Nbcons).
46. Montant global du chiffre d'affaires par jour (Datefacture, ca).
47. La liste des serveurs par nom et leur nombre de factures. Attention, les serveurs n'ayant fait aucune facture doivent apparaître dans le résultat (Nomserveur, Nbfactures).
48. La liste des serveurs par nom et le nombre de consommations qu'ils ont servies (Nomserveur, Nbcons).
49. La liste des serveurs par nom et leur chiffre d'affaire (somme des additions encaissées) (Nomserveur, ca).
50. Le nom des tables qui ont eu au moins deux factures (Nomtable, Nbfactures).
51. La liste complète des consommations et le nombre de factures dans lesquels elles apparaissent (Libcons, Nbfactures).
52. La liste complète des tables et leur chiffre d'affaires (Nomtable, ca).

Annexe B

Requêtes TP6

1. Donner la liste des noms, des prénoms et des dates de naissance de tous les élèves.
2. Donner tous les renseignements sur toutes les activités.
3. Lister les spécialités des professeurs.
4. Obtenir le nom et prénom des élèves pesant moins de 45 kilos et inscrits en $1^{\text{ère}}$ années ou élèves inscrits en $2^{\text{ème}}$ années.
5. Obtenir les nom des élèves dont le poids est compris entre 60 et 80 kilos.
6. Obtenir les nom des professeurs dont la spécialité est 'poésie' ou 'sql'.
7. Obtenir les nom des élèves dont le nom commence par 'L'.
8. Obtenir les nom des professeurs dont la spécialité est inconnue.
9. Obtenir les nom et prénom des élèves pesant moins de 45 kilos et inscrits en $1^{\text{ère}}$ année.
10. Obtenir pour chaque professeur, son nom et sa spécialité. Si cette dernière est inconnue, on souhaite afficher la chaîne de caractères : '****'.
11. Quels sont les noms et prénoms des élèves qui pratiquent du surf au niveau 1. Rédigez cette requête de cinq façons différentes.
12. Obtenir les nom des élèves de l'équipe AMC INDUS.
13. Obtenir les pairs de noms de professeurs qui ont la même spécialité.
14. Pour chaque spécialité sql/SQL, on demande d'obtenir son nom son salaire mensuel actuel et son augmentation mensuelle depuis son salaire de base.
15. Obtenir les nom des professeurs dont l'augmentation relative au salaire de bse dépasse 25%.
16. Afficher les points de Tsuno obtenus dans chaque cours sur 100 plutôt que sur 20.
17. Obtenir le poids moyen des élèves de $1^{\text{ère}}$ année.
18. Obtenir le total des points de l'élève numéro 3.
19. Obtenir la plus petite et la plus grande note de l'élève Brisefer.
20. Obtenir le nombre d'élèves inscrits en deuxième année.
21. Quelle est l'augmentation mensuelle moyenne des salaires des professeurs de SQL ?
22. Obtenir l'année de la dernière promotion du professeur Pucette.
23. Pour chaque professeur, afficher sa date d'embauche, sa date de dernière promotion ainsi que le nombre d'années écoulées entre ces deux dates.
24. Afficher l'âge moyen des élèves. Cet âge moyen sera exprimé en année.
25. Afficher les noms des professeurs pour lesquels il s'est écoulé plus de 50 mois entre l'embauche et la dernière promotion.
26. Obtenir la liste des élèves qui auront au moins 24 ans dans moins de 4 mois.
27. Obtenir la liste des élèves classée par année et par ordre alphabétique.
28. Afficher en ordre décroissant les points de Tsuno obtenus dans chaque cours sur 100 plutôt que sur 20.

29. Obtenir pour chaque élève de 1ère année son nom et sa moyenne.
30. Obtenir la moyenne des points de chaque élève de 1ère année dont le total des points est supérieur à 40.
31. Obtenir le maximum parmi les totaux de chaque élève.
32. Obtenirle nom des élèves qui jouent dans l'équipe AMC INDUS.
33. Quels sont les élèves de 1ère année dont la moyenne est supérieure à la moyenne de la 1ère année ?
34. Obtenir le nom et le poids des élèves de 1ère année plus lourds que n'importe quel élève de 2ème années.
35. Obtenir le nom et le poids des élèves de 1ère année plus lourds qu'un élève quelconque de 2ème années.
36. Obtenir le nom, le poids et l'année des élèves dont le poids est supérieur au poids moyen des élèves étant dans la même année d'études.
37. Obtenir les noms des professeurs qui ne donnent pas le cours numéro 1.
38. Obtenir les noms des élèves de 1ère année qui ont obtenu plus de 60% et qui jouent au tennis.
39. Professeurs qui prennent en charge tous les cours de deuxième année, on demande le Numéro et le nom.
40. Élèves qui pratiquent toutes les activités, on demande le Numéro et le nom.