Spécialité : INFORMATIQUE DÉCISIONNELLE

**Compte rendu du cours SQL**

Langage SQL (Structur Query Language) ou Langage de requête structurée en francais

C’est un langage adapter requêttage des donnée sous forme de Base de Données Relationnel, Sql n’est pas un langage impératif ou procédurale mais bien un langage déclaratif

A plusieurs responsabilités d’abord il organise son stockage de la meilleure manière possible pour obtenir les meilleurs résultats en termes de performance

Il assure aussi la cohérence des données

Il permet aussi l’exploitation de ces données

Permet aussi l’évolution des structures (ajouter modifier sup de colonne)

Contrôle d’accès permet l’accès à plusieurs utilisation au même temps au même donnée

On peut programmer avec plusieurs outille ou logiciel SGBDR = système de gestion de bases de données relationnelles (Oracl, Microsoft sql Server, My SQL)

Le SQL est séparé en différents sous-ensembles :

* -le **DML** est un langage qui permet de manipuler les données (SELECT, INSERT
* -**DDL** travaille sur la structure des tables, il permet de crée des objets, des vues…ou de les modifier c’est un langage qui permet de définir les structure.
* -le **DCL** permet d’attribuer des privilèges ou des droits au différents utilisateurs des Base de données
* Le **TCL** permet de gérer environnement transactionnel du moteur de Base de données

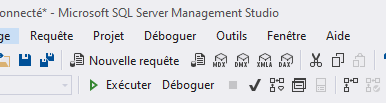
Le mécanisme (clé étrangère) de relation entre les tables permet d’économiser de l’espace, et assurer une sorte de contrainte (intégrité référentiel entre la table) exemple

Créer une table de « fournisseur » permet de gérer séparément les fournisseurs et de référencer partout avec des clés étrangères et ça permet aussi d’éviter les erreurs de saisie

Une base de données contient des tables reliées entre elle avec des associations

Ouvrir une nouvelle requête🡪s’assurer qu’on est dans la bonne base de données 🡪aller aux tables

L’exécution se fait avec le bouton exécuter ou bien la touche f5



Le FROM dans oracle est obligatoire

La majuscule et minuscule n’est pas important dans SQL Server management

**SELECT**

C’est une commande pour sélectionner des colonnes d’une table

Pour sélectionner plusieurs colonnes on sépare les colonnes par des virgules exemple :

SELECT Colonne1, colonne

FROM Table

**ALIAS**

Pour changer le titre de la colonne faut mettre le

SELECT nomPRECEDENT nomCHANGE

FROM Table

Ou bien avec la commande AS

SELECT nomPECEDENT AS nomCHANGE

FROM Table

Dans le Modèle Logique de Données, des base de donnée ( mysql,oracl…) les ligne et les colonne n’ont pas d’ordre, on peut permuter entre les lignes et entre les colonnes

SELECT \*

FROM Table est la syntaxe pour sectionner toute la table

**WHERE**

C’est une commande de restriction, elle permet de de sélectionner des lignes précises des informations désirer avec une syntaxe

SELECT Colonne1, colonne2, colonne3

FROM Table1

WHERE Condition1

Pour faire de la concaténation des valeurs des colonnes on fait

SELECT ‘Colonne1’ +’vide’+’colonne2’

FROM Table1

SELECT N’bonjour’

On rajoute le N pour stocker de Unicode qui est un codage moderne des chaines de caractère pour stocker tous les caractères du monde.

La commande ORDER BY (trier par) sert à trier, et le trie se fait par ordre alphabétique

SELECT colonne1, colonne2

FROM table

ORDER BY colonne1

CREATE est une commande qui sert à créer une table comme la syntaxe l’indique

CREATE TABLE nom\_de\_la\_table

(

colonne1 type\_donnees,

colonne2 type\_donnees,

colonne3 type\_donnees,

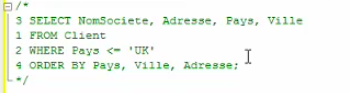
colonne4 type\_donnees

)

GO entre CREATE ET l’instruction qui va suivre sert à laisser le process (la requête) de création le temps de finir la création puis passer à l’’instruction qui suit

Utiliser un alias permet de renommer le nom d’une colonne dans un résultat d’une requête pour avoir un nom facilement à l’utilisateur

L’ordre du d’exécution des instructions des commandes se fait selon un ordre suivant

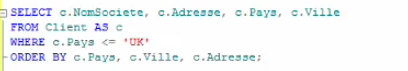


Il sélectionne la table🡪sélectionne les lignes et les conditions de restriction 🡪sélectionne la colonne à afficher🡪en fin il fait le tri par ordre alphabétique.

LORS DE LALIAS si on met notre ALIAS dans le SELECT et on fait ORDER BY ALIAS 🡺on va avoir une erreur car dans l’ordre d’exécution car le SELECT s’exécute après le ORDER BY.

**Les alias**

L’alias sert à renommé (les tables, les colonnes …) sert à abréger les noms des tables …. Exemple : alias Table client en C



Exemple alias Colonne nomSociete en Nom



On essaye toujours de mettre des alias claire et compréhensible par le développeur ou collègues qui vont lire notre code.

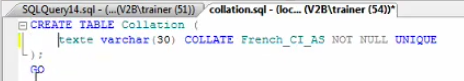
Pour chaque colonne de table on met ALIAS.Colonne pour différencier entre les colonnes des tables

**Les collations :**

Collation est un concept qui permet de maitriser de filtre du WHERE et le trie de ORDER BY

Elle est introduite dans la base de données pour palier à des problématiques propres a du système informatique et la gestion de la chaine de caractère (l’ordre informatique ou ASCII) par des tables de correspondance qui permet de dire exemple le A = a = à =â et comme ça on pourra trier par un ordre plus compréhensible pas l’humain.

On a un exemple de la collation français

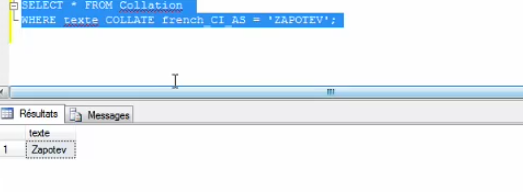


Frensh = le code de page, les caractères compris par cette collation

CI = insensible à la casse (majuscule et minuscule

AS = sensible aux accents elle peut être aussi AI (insensible aux accents)

COLLATE est une clause qui nous permet de changer dynamiquement pendant la requête la collation

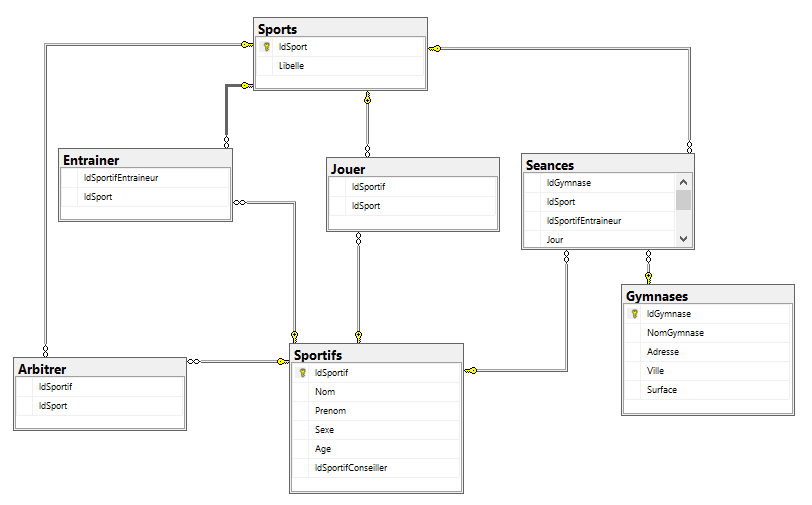


Le résultat envoi une seule valeur qui correspond à la condition de la COLLATE que nous avons (insensible à la casse et sensible aux accents)

**Les schémas**

Le schéma c’est comme un arbre généalogique d’une famille mais d’une base de données.

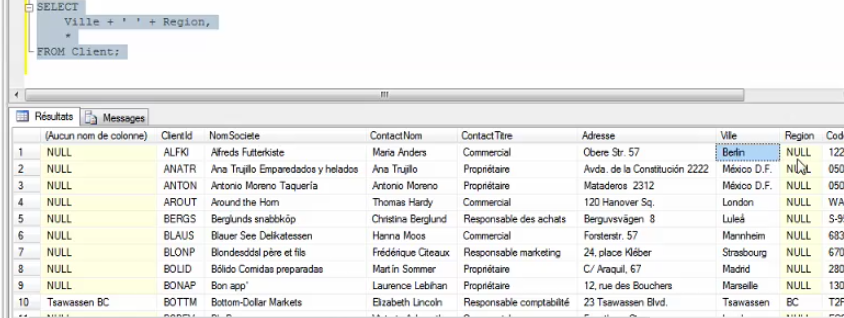
TP GYMNASE :



La gestion des NULL :

Le NULL est n’est pas une valeur, c’est un marqueur indicateur que la valeur est inconnu

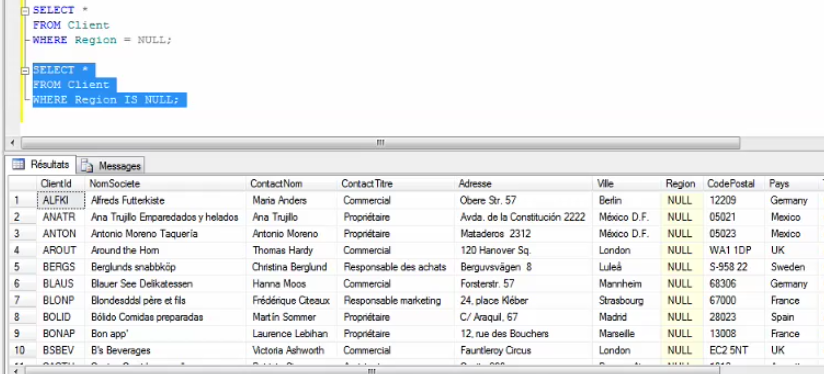
On ne peut pas faire de comparaison avec NULL ni quelconque opération mathématique avec d’autre valeur ni Concaténer une valeur connue avec une valeur inconnu NULL



La concaténation d’une colonne ville connu avec la colonne région inconnu revoit une valeur inconnu NULL

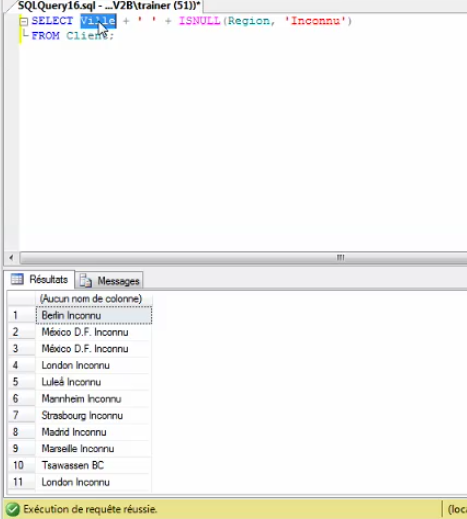
Le marqueur NULL dans la norme SQL colonise grignote /s’entends sur la totalité de l’opération

Pour sélection les NULL on utilise l’instruction « IS  NULL» exemple :



Quand on met la condition de la clause WHERE REGION IS NULL résultat nous revoie tous les NULL

On peut quand même retourner une valeur lors de la concaténation entre une colonne connu et une autre inconnu à l’aide d’une **fonction ISNULL** comme dans l’exemple suivant :



SELECT COLLONE1 + ‘ ’ + ISNULL (PARAMETRE1, ’parametre2’)

Si le premier paramètre est NULL alors il revoit le deuxième parametre2

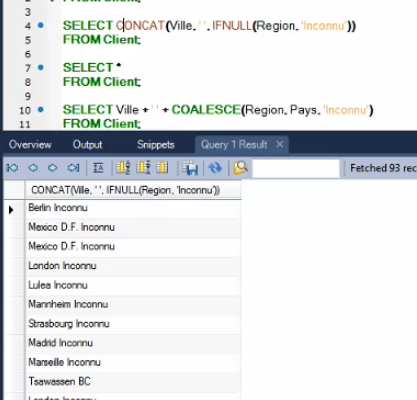
Dans microsft sql serveur

**ISNULL** = fonction limitée à un seul paramètre et qui permet de sélectionner les NULL

**COALESCE** = est une fonction comme ISNULL avec des paramètres en plus à ajouter

Dans mySQL ISNULL devient IFNULL

Et pour faire la concaténation le + devient .



SELECT CONCAT (COLLONE. ’ ’ .IFNULL (parametre1.’parametre’))

FROM TABLE

Si le premier paramètre est NULL alors on Revoit le deuxième paramètre.

SELECT COLONNE. ’ ’ .COALESCE (parametre1.’parametre2’.’Parametre 3))

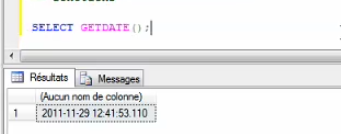
FROM TABLE

1. **Les fonctions**
2. **Fonction sur les dates**

Sont des mécanismes pensent dans tous les langages de programmations

La fonction est le nom d’une routine d’un sous-programme

La commande **GETDATE** Pour afficher la date et l’heure courante



Ou bien CURRENT\_TIMESTAMP une fonction qui correspond a la norme SQL,

**NB : il faut s’habituer à utiliser les fonctions de la norme SQL pour que notre code soit valable dans les différents moteurs**

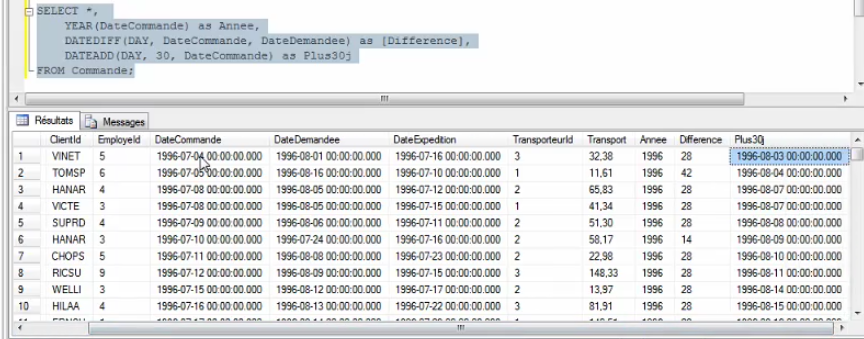
**La fonction TIMESTAMP()** : elle est utilisée dans le langage SQL pour obtenir un DATETIME à partir d’une DATE.

**La Syntaxe :**

SELECT TIMESTAMP("YYYY-MM-DD"); -- résultat sous la format YYYY-MM-DD HH:MM:SS

SELECT TIMESTAMP("YYYY-MM-DD", "HH:MM"); -- résultat sous la format YYYY-MM-DD HH:MM:SS

On peut manipuler les dates avec différentes fonctions exemple



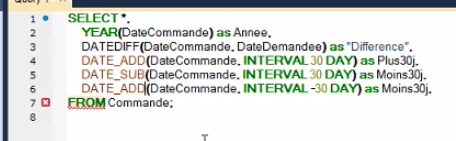
YEAR (datecommande) as Année : renvoie l’année de la date de commande dans un alias Année.

DATEDIFF( day, datecommande, datedemande) as Différence : renvoie la différence entre la date de commande et la date de demande en unité DAY /jour dans un alias Différence.

DATEADD(day ,30 , datecommande) as plus30j : permet d’ajouter 30 jour à la date commande.

DADEPART( month ,datecommande) as mois : permet d’extraire le mois ou autre partie d’une date

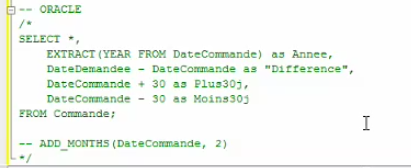
Dans MySQL les syntaxes sont un peu différentes on va voir quelques exemples.



DATE\_ ADD pour ajouter des jours /mois /années

DATE\_SUB pour soustraire des jours /mois/ années

Et différente aussi avec la syntaxe dans ORACLE



1. **Fonction chaine de caractères**

# CHARINDEX ()

C’est une fonction de recherche dans une chaine de caractère

**La Syntaxe :**

CHARINDEX ( ‘recherche’,’ colonne’)

FROM Table ;

Pour prendre une valeur à gauche on utilise LEFT exemple :

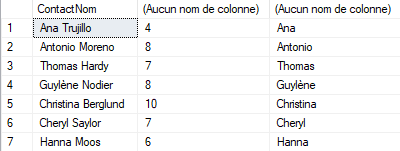
On a une colonne de nom prénom dans une seule colonne et on veut les séparer

select ContactNom ,

charindex(' ',ContactNom), // on cherche l’espace

LEFT (ContactNom ,charindex(' ' ,ContactNom))//on prends la chaine a gauche de l’espace.

from Client



**On cherche la position de l’espace qui sépare le nom et le prénom dans la colonne contactnom avec un CHARINDEX (‘ ‘, contacteNom ),**

1. **Puis on renvois la valeur à gauche de cette espace avec la fonction LEFT qui est le prénom.**

# LEN (colonne)

LA FONCTION **LEN (colonne)** renvoie la longueur de la chaine de caractères dans SQL Server

Et **LENGTH ()** dans MySQL et Oracle

**La syntaxe :**

SELECT LENGTH ('exemple');

# SUBBSTRING ()

La FONCTION **SUBBSTRING ()** renvoi un sous ensemble d’une chaine

**La syntaxe :**

SUBSTRING (chaine, début, longueur)

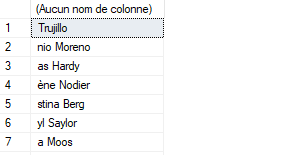
Retourne la chaîne de caractère “chaine” en partant de la position définie par “début” et sur la longueur définie par “longueur”

**Exemple :**

SELECT SUBSTRING (ContactNom, 5, 10)

FROM client

Résultat



# CHARINDEX () :

Cette fonction recherche une expression de caractères à l’intérieur d’une deuxième expression de caractères, retournant la position de départ de la première expression si elle est trouvée

**La syntaxe**

CHARINDEX (expressionToFind, expressionToSearch, [start\_location ] )

Exemple :

SELECT CHARINDEX ('t', 'TechOnTheNet.com', 8);

Résultat :



# TRIM ()

Dans le langage SQL la fonction **TRIM ()** permet de supprimer des caractères au début et en fin d’une chaîne de caractère. Pour enlever les espace au début de la chaine on utilise la commande **LTRIM ()** qui fait Left Trim (une suppression des caractères au début d’une chaine). Et **RTRIM ()** Right Trim permet de supprimer des caractères à la fin d’une chaîne de caractères.

**La syntaxe :**

SELECT TRIM ('   Colonne   ')

# REPLACE ()

La fonction REPLACE dans le langage SQL permet de remplacer des caractères alphanumérique dans une chaîne de caractère. Cela sert particulièrement à mettre à jour des données

La fonction comporte 3 paramètres :

1. Chaîne d’entré
2. Texte à remplacer
3. Texte qui sert de remplacement

**Exemple  simple:**

SELECT REPLACE ('Hello tout le monde', 'Hello', 'Bonjour');

Résultat :



**Fonction REPLACE dans un UPDATE**

UPDATE table

SET nom\_colonne = REPLACE (nom\_colonne, 'ancien texte', 'texte de remplacement')

**Fonction REPLACE dans un SELECT :**

SELECT colonne1, colonne2, REPLACE (colonne3, 'ancien texte', ' texte de remplacement')

FROM table

# CONCAT ()

Dans le langage SQL la fonction CONCAT () permet de concaténer les valeurs de plusieurs colonnes pour ne former qu’une seule chaîne de caractère.

**La syntaxe  simple:**

CONCAT (colonne1, colonne2 )

**La syntaxe dans un SELECT :**

SELECT id, CONCAT (colonne1, colonne2)

FROM `table`

**La syntaxe dans une clause WHERE:**

SELECT id

FROM `table`

WHERE colonne1 = CONCAT (colonne2, colonne3)

# UPPER ()

La fonction UPPER () dans le langage SQL permet de transformer tous les caractères en minuscules d’une chaîne de caractère en majuscules

# LOWER ()

La fonction LOWER () dans le langage SQL permet de transformer tous les caractères d’une chaîne de caractère en minuscules

1. **Fonction Numérique**

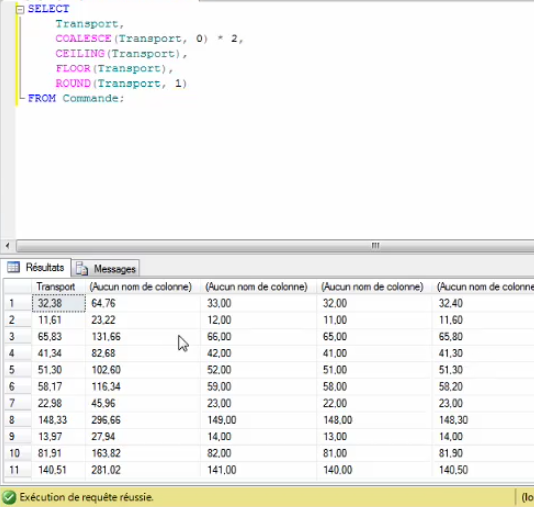
Si dans la structure de la colonne le NULL est permis alors on doit se méfier et prendre nos précaution des NULL en utilisant COALESCE ( ColonneNull,0)

# CEILING(Colonne)

Pour arrondir une valeur décimale max (le plafond) en un entier 12.32🡪13

# FLOOR(Colonne)

Pour arrondir une valeur décimale min (le planché) 12.67 🡪12



# ROUND ( colonne, N)

C’est une fonction qui arrondit un décimal à N chiffre après la virgule.

Lors de la division le résultat renvoi le même type que la valeur précise

Float/int = float

# CAST()

CAST (Colonne as décimal(x,y) est une norme SQL et fonction de conversion de type

X c’est le nombre total de chiffre à renvoyer

Y c’est le nombre de chiffre après la virgule comprise dans x

Ces fonctions numériques sont les même dans MYSQL et dans ORCLE juste pour CEILING🡪 CEIL dans oracle

1. **Technique avancée du SELCT**

Le moteur a deux façons de retrouver des données

* Soit il parcourt la table ligne par ligne (sans indexe) on appelle ça un parcours ou scan de table 🡪le moteur SQL va perdre des performances au fur et à mesure et la table va prendre en table de ligne
* On utilise les indexes 🡪il va aller chercher dans la table des indexe

# L’opérateur BETWEEN

Il permet de sélectionner un intervalle de données dans une requête utilisant WHERE. L’intervalle peut être constitué de chaînes de caractères, de nombres ou de dates

**La syntaxe :**

SELECT \*

FROM table

WHERE colonne BETWEEN 'valeur1' AND 'valeur2'

Exemple :

On veut sélectionner toute les dates de commande comprise entre le 01/07/1996 et 31/07/1996

Et dans notre exemple on spécifie l’heure car on peut avoir une commande qui a été faite le 31/07/1966 après minuit, donc on veut s’assurer qu’on toute les date de commande du mois de juillet 1996



# L’opérateur LIKE ()

Il est utilisé dans la clause WHERE et permet d’effectuer une recherche sur un modèle particulier. Il est par exemple possible de rechercher les enregistrements dont la valeur d’une colonne commence par telle ou telle lettre.

**La Syntaxe :**

SELECT \*

FROM table

WHERE colonne LIKE modèle

Exemple : on veut chercher tous les prénoms qui commencent par « An »



Le % est considéré comme joker qui remplace tous les autres caractères.

**• LIKE ‘%a’ :** permet de rechercher toutes les chaines de caractère qui se termine par un « a ».

**• LIKE ‘a%’ :** permet de rechercher toutes les lignes de « colonne » qui commence par un « a ».

• **LIKE ‘%a%’ :** permet de rechercher tous les enregistrements qui utilisent le caractère « a ».

• **LIKE ‘pa%on’ :** permet de rechercher les chaines qui commence par « pa » et qui se terminent par « on », comme « pantalon » ou « patron ».

• **LIKE ‘a\_c’ :** peu utilisé, le caractère « \_ » (underscore) peut être remplacé par n’importe quel caractère, mais un seul caractère uniquement (alors que le symbole pourcentage « % » peut être remplacé par un nombre incalculable de caractères.

# CASE()

Dans le langage SQL, la commande « CASE … WHEN … » permet d’utiliser des conditions de type « si / sinon » (cf. if / else) similaire à un langage de programmation pour retourner un résultat disponible entre plusieurs possibilités. Le CASE peut être utilisé dans n’importe quelle instruction ou clause, telle que SELECT, UPDATE, DELETE, WHERE, ORDER BY ou HAVING.

L’utilisation du CASE est possible de 2 manières différentes :

**La Syntaxe :** Comparer une colonne à un set de résultat possible

CASE Colonne

WHEN 1 THEN 'un'

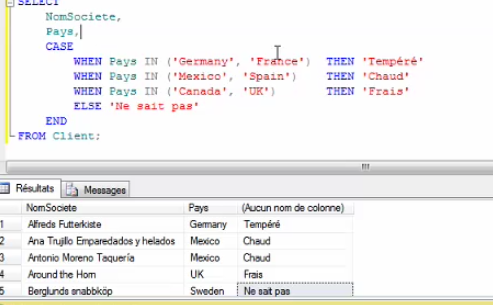
WHEN 2 THEN 'deux'

WHEN 3 THEN 'trois'

ELSE 'autre'

END

**Exemple :** On veut faire classer les pays selon leurs climats



**La Syntaxe :** Élaborer une série de conditions booléennes pour déterminer un résultat

CASE

WHEN a=b THEN 'A égal à B'

     WHEN a>b THEN 'A supérieur à B'

  ELSE 'A inférieur à B'

END

**Exemple :**



1. **Les jointures**

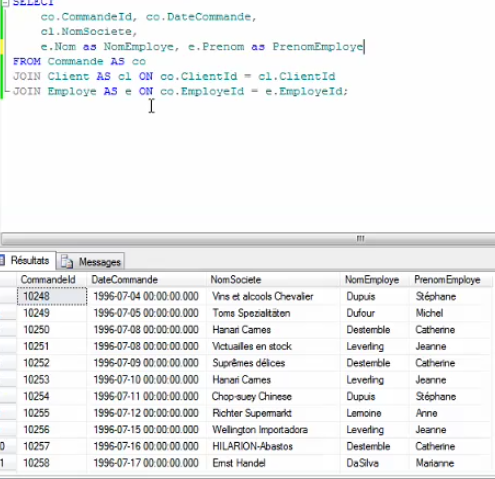
Les jointures en SQL permettent d’associer plusieurs tables dans une même requête. Cela permet d’exploiter la puissance des bases de données relationnelles pour obtenir des résultats qui combinent les données de plusieurs tables de manière efficace

**La Syntaxe**

SELECT \*

FROM Colonne1

JOIN Colonne2 ON Condition.colonne 1= Condition.colonne 1

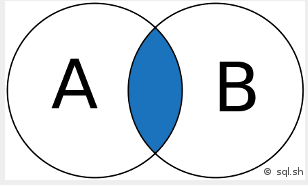


Dans notre exemple on fait une jointure entre les table commande et client et commande et employé.

Il existe plusieurs

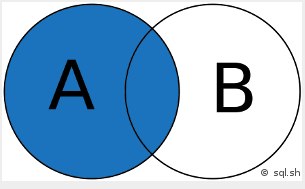
# INNER JOIN :

C’est une jointure interne pour retourner les enregistrements quand la condition est vraie dans les 2 tables. C’est l’une des jointures les plus communes.



# LEFT JOIN (ou LEFT OUTER JOIN) :

Cest une jointure externe pour retourner tous les enregistrements de la table de gauche (LEFT = gauche) même si la condition n’est pas vérifié dans l’autre table.



Exemple :

/\*les entraineurs qui sont aussi joueurs \*/

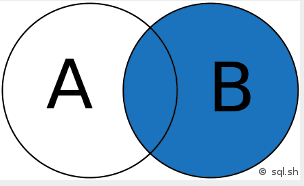
select \*

from Entrainer e

left join Jouer j on j.IdSportif=e.IdSportifEntraineur ;

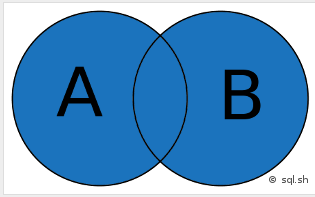
# RIGHT JOIN (ou RIGHT OUTER JOIN) :

C’est une jointure externe pour retourner tous les enregistrements de la table de droite (RIGHT = droite) même si la condition n’est pas vérifié dans l’autre table.



# FULL JOIN (ou FULL OUTER JOIN) :

C’est une jointure externe pour retourner les résultats quand la condition est vrai dans au moins une des 2 tables



# CROSS JOIN :

C’est une jointure croisée permettant de faire le produit cartésien de 2 tables. En d’autres mots, permet de joindre chaque ligne d’une table avec chaque ligne d’une seconde table. Attention, le nombre de résultats est en général très élevé.

# SELF JOIN :

Effectuer une jointure d’une table avec elle-même comme si c’était une autre table.

# NATURAL JOIN :

Jointure naturelle entre 2 tables s’il y a au moins une colonne qui porte le même nom entre les 2 tables SQL.

**Cas spécial**

Faire une jointure externe suivie d’une jointure interne nous renvoie rien car la jointure externe renvoie des NULL et ces NULL n’ont pas de correspondance avec d’autre ligne de la table joint

Pour pallier à ce problème que ça revoie il faut mettre une jointure externe (LEFT) après jointure externe (LEFT)

Ou bien faire une jointure interne à l’intérieur d’une jointure externe.

1. **Operateurs ensembliste**

# 1-L’operateur UNION :

N’a pas le droit de renvoyer des doublons et c’est un opérateur ensembliste il réagit selon les règles du model relationnel et la théorie des ensemble.

C’est donc une commande qui permet de concaténer les résultats de 2 requêtes ou plus. Pour l’utiliser il est nécessaire que chacune des requêtes à concaténer retournes le même nombre de colonnes, avec les mêmes types de données et dans le même ordre.

**La syntaxe :**

SELECT \* FROM table1

UNION

SELECT \* FROM table2

La commande UNION ne renvoie pas de doublant dans le résultat.

**Exemple :**

select sf.Nom from Sportifs Sf

union

select s.Libelle from Sports S

resultat :



On fait une concaténation de deux colonnes de deux tables différentes.

Contrairement à UNION ALL qui renvoi toute les lignes même les doublant avec la même syntaxe que celle d’UNION

# 2-L’operateur INTERSECT :

permet d’obtenir l’intersection des résultats de 2 requêtes. Cette commande permet donc de récupérer les enregistrements communs de 2 tables.

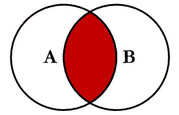
**La Syntaxe**

SELECT \* FROM table A

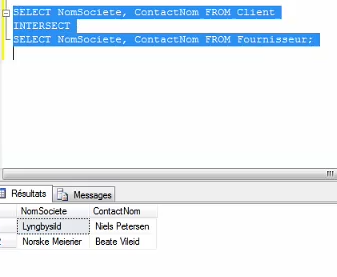
INTERSECT

SELECT \* FROM table B

Schéma :



Exemple : on veut renvoyer l’intersection de deux tables à deux colonnes similaires.



# 3-L’Operateur EXCEPT

s’utilise entre 2 instructions pour récupérer les enregistrements de la première instruction sans inclure les résultats de la seconde et leurs intersections dans le résultat final.

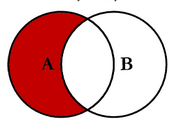
**La syntaxe**

SELECT \* FROM table1

EXCEPT

SELECT \* FROM table2

Schéma :



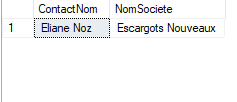
Exemple : on renvoi tous les fournisseurs qui ne sont pas client.

select f.ContactNom,f.NomSociete from Fournisseur f

except

select c.ContactNom,c.NomSociete from Client c

resultat :



**Les SOUS-REQUETE :** il existe deux types

# Sous requête simple

La sous-requête (aussi appelé « requête imbriquée » ou « requête en cascade ») consiste à exécuter une requête à l’intérieur d’une autre requête

Le select interne s’exécute une fois et en premier pour avoir **une** valeur que le select externe vient exécuter en second.

Exemple :

select ContactNom

from ( select \* from client ) nom

resulat :



# Sous requête impriqué

Une requête imbriquée est souvent utilisée au sein d’une clause WHERE ou de HAVING pour remplacer une ou plusieurs constante.

**La syntaxe**

SELECT \*

FROM table

WHERE colonne = (SELECT Colonne FROM Table WHERE Condition)

Exemple : sélectionner tous clients qui habitent dans la même ville que le client 'PARIS'

select \*

from Client

where ville = (select Ville from client where ClientId = 'PARIS' );

Résultat :



# Sous requête corrélé

La requête SELECT interne prends une valeur de la requête externe pour s’exécuter donc elle a un effet boucle. Requête externe puis interne puis externe…

Exemple :

C:\Users\PERSO\Desktop\capture SQL\sousReqCOR.PNG

On ne peut pas utiliser une sous requête corrélé à l’intérieur de la clause FROM

Soit on l’utilise dans le SELECT ou dans la close WHERE

# L’operateur EXISTS

consiste à vérifier si une sous-requête retourne un résultat ou non, en utilisant EXISTS dans la clause conditionnelle (where,…)

Exemple : on veut avoir les clients qui n’ont pas passé de commande

select \*

from Client cl

where not exists ( select \* from Commande co where co.ClientId=cl.ClientId)

Résultats :



# Operateur ALL :

permet de comparer une valeur dans liste de valeurs d’une sous-requête. En d’autres mots, cette commande permet de s’assurer qu’une condition est “égale”, “différente”, “supérieure”, “inférieure”, “supérieure ou égale” ou “inférieure ou égale” pour **tous** les résultats retourné par une sous-requête.

Exemple : on veut sélectionner tous les clients qui n’ont pas fait de commande

select \*

from client

where ClientId <> all (select ClientId from commande)

Résultats



# Operateur ANY

réponds a la question de y’aurai-t-il au moins un

Exemple: sélectionner n’importe quel employé qui a un salaire plus élevé qu’un commercial

select \*

from employees e1

where e1.salary > any (select e2.salary from employees e2

join jobs j on e2.job\_id=j.job\_id

where j.job\_title = 'Administration Assistant')

Résultat :



1. **Fonction d’agrégation :**

# Fonction d’agrégation :

Les fonctions d’agrégation dans le langage SQL permettent d’effectuer des opérations statistiques sur une liste de valeur.

## Count () permet de compter le nombre de lignes d’une colonne

## Max (colonne) pour avoir la grande valeur d’une colonne

## Min (colonne) pour avoir la plus petite valeur d’une colonne

## Sum(colonne) faire la somme des valeur numérique

## Avg(colonne) pour calculer la moyenne

Exemple :

select count(\*) Nombre\_Commande,

max(DateCommande) derniere\_date\_commande,

min(datecommande) premiere\_date\_commande,

sum(transport) somme\_cout\_trans,

avg(transport) moyenne\_cout\_trans

from commande

Résultat :



**Exemple**: d’un COUNT(Region )

select count(\*),

count(distinct region) sans\_doubl,

count(region)

from Client

Résultat :



Count (Region) compte le nombre de ligne d’une colonne avec les NULL

Count (DISTINCT Region) nous renvoi le nombre de ligne de Région sans doublant ou sans NULL

On ne peut pas avoir à la fois des détails et des résultats il faut faire des fonctions d’agrégation

# LE GROUPE BY

Pour grouper plusieurs résultats et utiliser une fonction de totaux sur un groupe de résultat.

**La Syntaxe :**

SELECT colonne1, fonction (colonne2)

FROM table

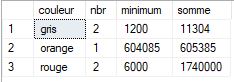
GROUP BY colonne1

Exemple :

select couleur,count(\*) nbr,min(prix) minimum,sum(cout) somme

from VOITURE

group by couleur



L’ajout de la commande « **WITH ROLLUP** » permet quant à elle d’ajouter une ligne supplémentaire qui fonctionne tel qu’un système de “super-agrégateur” sur l’ensemble des résultats.

**La syntaxe :**

SELECT colonne1, fonction(colonne2)

FROM table

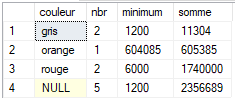
GROUP BY colonne1 WITH ROLLUP

Exemple :

select couleur,count(\*)nbr,min(prix) minimum,sum(cout) somme

from VOITURE

group by couleur with rollup



**Cas Special : pour trouver les doublants**

**La syntaxe :**

SELECT   COUNT(\*) AS nbr\_doublon, champ1, champ2, champ3

FROM     table

GROUP BY champ1, champ2, champ3

HAVING   COUNT(\*) > 1

Exemple :

select couleur, prix, marque, count(\*)

from VOITURE

group by couleur, prix, marque

having count(\*) >1



# HAVING

La condition HAVING en SQL est presque similaire à WHERE à la seule différence que HAVING permet de filtrer en utilisant des fonctions d’agrégation.

**La syntaxe :**

SELECT colonne1, SUM (colonne2)

FROM nom\_table

GROUP BY colonne1

HAVING fonction (colonne2) operateur valeur

# LIMIT

La clause LIMIT est à utiliser dans une requête SQL pour spécifier le nombre maximum de résultats que l’on souhaite obtenir.

Elle est associée à un OFFSET, c’est-à-dire effectuer un décalage sur le jeu de résultat.

**La Syntaxe simple**

SELECT \*

FROM table

LIMIT 10

## La Syntaxe Offset avec PostgreSQL

SELECT \*

FROM table

LIMIT 10 OFFSET 5

## La Syntaxe Offset avec MySQL :

SELECT \*

FROM table

LIMIT 5, 10

# L’ordre d’évaluation

1-exacuter le FROM

2-on filtre par le WHERE

3-on crée des sous-ensembles avec le GROUB BY

4- on filtre encore avec HAVING

5-on envoi les résultats et affiche avec SELECT

6- on le trie *avec le ORDER* BY

# Les modifications de données

Il existe 3 types de modification

1. Ajout de donnée à une table à l’aide de l’instruction INSERT
2. La modification de donnée à l’aide de l’instruction UPDATE
3. la suppression de donnée à l’aide de l’instruction DELETE

Une instruction transactionnel ou atomique est une instruction ou considéré comme un tous. C’est quand une suppression se porte sur exemple 10 ligne et qu’une violation de contrainte se produit à la 6 ligne alors les 5premiere sont pas supprimé mais mise en attente (on dit aussi un mode autocomit)

Il existe un mode dans oracle pour confirmer ou valider la suppression il faut le faire manuellement avec un commit

1. **L’insertion de données**

# ALTER TABLE

La commande ALTER TABLE en SQL permet de modifier une table existante. Idéal pour ajouter une colonne, supprimer une colonne ou modifier une colonne existante, par exemple pour changer le type.

**La Syntaxe : Pour l’ajout d’une colonne**

ALTER TABLE nom\_table

ADD Nom\_Colonne Type\_Colonne

**La Syntaxe : Pour la supression d’une colonne**

ALTER TABLE nom\_table

DELETE Nom\_Colonne Type\_Colonne

# La commande INSERT INTO

L’insertion de données dans une table s’effectue à l’aide de la commande INSERT INTO. Cette commande permet au choix d’inclure une seule ligne à la base existante ou plusieurs lignes d’un coup.

**La Syntaxe :**

INSERT INTO table (colonne1,colonne2….)

 VALUES ('valeur 1', 'valeur 2', ...)

**Exemple :** insertion de plusieurs ligne à la fois dans la table Editeur qui a des colonne Nom, Téléphone, Pays les valeur suivant :

nsert into Editeur ( Nom,Telephone,Pays)

values ('meriame','0677551909','france'),

('nadia','0612155236','espagne'),

('Dyhia','0766805704','algerie'),

('lynda','0670131504','france'),

('samir','0612138004','italie'),

('samir',NULL,NULL);

Il faut faire attention aux colonnes NON NULL qui n’accepte pas de valeur NULL où le champ est obligatoire à remplir

# Une colonne identité

**Dans Microsoft SQL Server** Une colonne identité est une colonne auto-incrémentale c’est une colonne qui contient dans sa définition l’information que elle va être elle-même gérer au niveau de ses numéros, et les valeurs vont être générer par le moteur de base de données.et on l’a declare lors de la création des table comme suit :

CREATE TABLE Editeur

(

IdEditeur INT NOT NULL identity, -- identifiant auto-incrément

Nom VARCHAR (30),

Telephone VARCHAR(15),

Pays VARCHAR(30),

CONSTRAINT PK\_Editeur primary key (IdEditeur)

);

**Dans MySQLl** aussi existe les colonnes auto incrémentale mais on peut aussi forcer l’ajout et c’est un peu risqué, car on peut bien ajouter des valeurs qui existe ou n’existe pas

**Dans ORACLE** utilise un mécanisme de séquenceur un objet qui va gérer lui-même l’incrémentation de numéro

# REPLACE ()

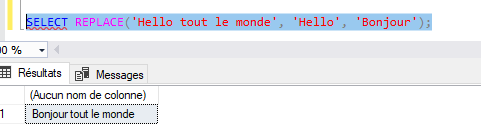
La fonction REPLACE permet de remplacer des caractères alphanumérique dans une chaîne de caractère. Cela sert particulièrement à mettre à jour des données dans une base de données ou à afficher des résultats personnalisés.

**La Syntaxe :**

SELECT REPLACE (colonne ou ‘valeur’, x , y)

On remplace x par y dans la colonne en affichage

Exemple : on va remplacer hello dans hello tout le monde par bonjour



# REPLACE dans un UPDATE

Pour modifier certains caractères dans une base de données, on utiliser :

**La Syntaxe :**

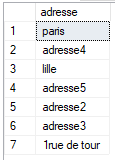
UPDATE table

SET nom\_colonne = REPLACE (nom\_colonne, 'ancien texte', 'texte de remplacement')

**Exemple :**

update CLIENT

set adresse= replace (adresse,'paris','Lyon')



# REPLACE dans un SELECT

Pour ne remplacer les caractères que lors de l’affichage et ne pas altérer les données contenu dans la table, on utlise :

**La sytaxe :**

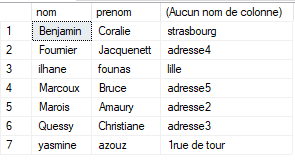
SELECT colonne1, colonne2, REPLACE(colonne3, 'exemple insulte, 'CENSURE')

FROM table

**Exemple :**

select nom,prenom, replace (adresse,'lyon','strasbourg')

from CLIENT



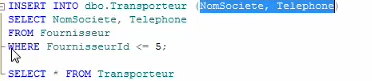
A voir dans Transact SQL

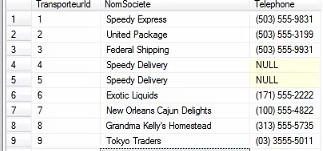
La fonction scope\_identity () Elle retourne la dernière valeur insérée par l’insertion précédente ou bien @@identity c’est une variable système, elle est moins sûr car elle pourra être modifié par du code qui serait appelé par l’insert qu’on appelle déclencheurs



# Insertion de données avec le select

Elle permet de générer des donnes à partir d’autre table en utilisant la syntaxe et l’exemple suivant ou on veut ajouter /copier les colonnes (NomSociete, Téléphone) de la table fournisseur dans la table transporteur





Et dans le résultat on voit bien que des fournisseurs tel que EXOTIC LIQUIDS …ont été ajouté à la table fournisseur

L’OPERATEUR **UPPER ()** permet de transformer des minuscule en majuscule exemple

C:\Users\PERSO\Desktop\capture SQL\upper.PNG

On voit dans le résultat que les valeurs de notre colonne sont passées de la minuscule en majuscule

# LEFT()

Pour prendre les premiers caractères d’une colonne on utilise la fonction LEFT

**La Syntaxe :**

SELECT

LEFT (colonne, nombre de char)

From Table

1. **La mise à jour des données**

# La commande UPDATE

Elle permet d’effectuer des modifications sur des lignes existantes. Très souvent cette commande est utilisée avec WHERE pour spécifier sur quelles lignes doivent porter la ou les modifications.

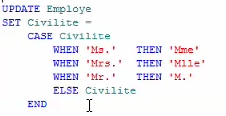
La mise à jour de données existante se fait à l’aide de la commande update qui a 3 parties

UPDATE Table

SET Colonne à modifier = La Modification

**Exemple :**

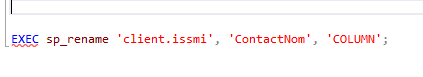
C:\Users\PERSO\Desktop\capture SQL\update1.PNG 

On veut changer la civilité anglaise en civilité française

Pour modifier le nom d’une colonne dans SQL Server on utilise EXEC sp\_Rename

**la syntaxe :**



1. **La suppression de données**

# La commande DELETE

permet de supprimer des lignes dans une table. En utilisant cette commande associée à WHERE il est possible de sélectionner les lignes concernées qui seront supprimées.

**La Syntaxe :**

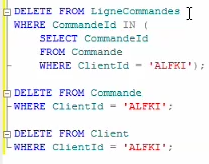
DELETE FROM table

WHERE condition

Et en oubliant de spécifier la clause WHERE on risque de supprimer toute la table.

On ne peut pas supprimer une ligne si elle est référencée dans une autre table (clé étrangère) et pour éviter la violation de cette contrainte (FK) on doit supprimer la valeur d’une table après une autre dans le sens des clés étrangères.

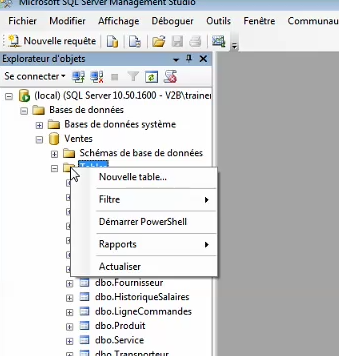
Exemple :



**CREATION DES TABLE :**

La création de table se fait graphique et en ligne de commande.

Graphiquement on suit les étapes suivantes



**2**

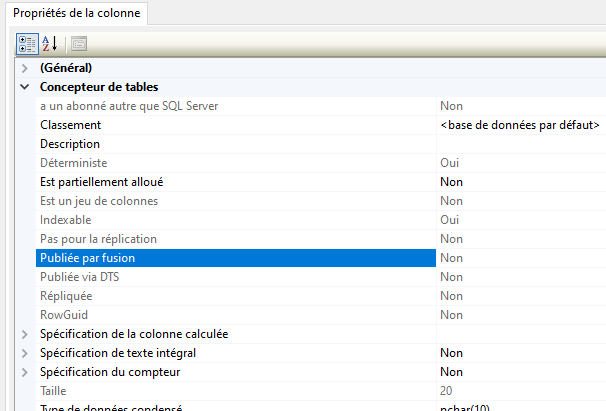
**1**

Ensuit les étapes suivantes :

1. Clic droit TABLE
2. Nouvelle TABLE



1. Introduire le nom de la table et son type et si on autorise les marqueurs NULL
2. Puis on a les propriétés des colonnes ou on a plusieurs paramètres.



# la fonction CREATE () :

Pour créer une base de données :

create database TP2\_GYM

Pour créer une table :

CREATE TABLE Organisme

(

IdOrganisme INT NOT NULL identity ,

Nom VARCHAR (50),

Telphone\_Organisateur VARCHAR(50),

Pays\_Organisateur VARCHAR(50),

CONSTRAINT PK\_Organisme primary key (IdOrganisme), -- déclarer une clé primaire

);

Pour éviter d’avoir une erreur en cas où la BD existe déjà ?

IL faut ajouter « IF NOT EXISTS »:

CREATE TABLE iF NOT EXISTS Organisme

# MERGE

 La commande MERGE permet d’insérer ou de mettre à jour des données dans une table. Cette commande permet d’éviter d’effectuer plusieurs requêtes pour savoir si une donnée est déjà dans la base de données et ainsi adapter l’utilisation d’une requête pour ajouter ou une autre pour modifier la donnée existante

Sa **syntaxe** est :

MERGE INTO table1

USING table\_référence

ON (conditions1)

WHEN MATCHED THEN

UPDATE SET table1.colonne1 = valeur1, table1.colonne2 = valeur2

DELETE WHERE conditions2

WHEN NOT MATCHED THEN

INSERT (colonnes1, colonne3)

VALUES (valeur1, valeur3)

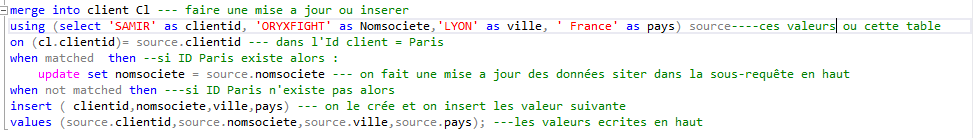
1-MERGE INTO : permet de sélectionner la table à modifier

2-USING et ON : permet de lister les données sources et la condition de correspondance clause ON

3-WHEN MATCHED : permet de définir la condition de mise à jour lorsque la condition est vérifiée

4-WHEN NOT MATCHED : permet de définir la condition d’insertion lorsque la condition n’est pas vérifiée

Exemple : l’explication dans les commentaires



# Transaction

Pour éviter l’incohérence dans nos tables et éviter qu’une des lignes soit incomplète exemple si on supprime une partie d’une table

Et la transaction regroupe plusieurs requêtes

Une transaction complète doit assurer 4propriete qu’on appelle l’acidité de la transaction (**ACID**), ce sont des critères que cette transaction doit impérativement respecter.

**Atomique** : assure qu'une transaction se fait au complet ou pas du tout

**Cohérente** : La propriété de cohérence assure que chaque transaction amènera le système d'un état valide à un autre état valide

**Isolé** : La propriété d'isolation assure que l'exécution simultanée de transactions produit le même état que celui qui serait obtenu par l'exécution en série des transactions. Si T1 et T2 s'exécutent simultanément, alors chacune doit demeurer indépendante de l'autre.

**Durable** : Assure que lorsqu'une transaction a été confirmée, elle demeure enregistrée même à la suite d'une panne d'électricité, d'une panne de l'ordinateur ou d'un autre problème

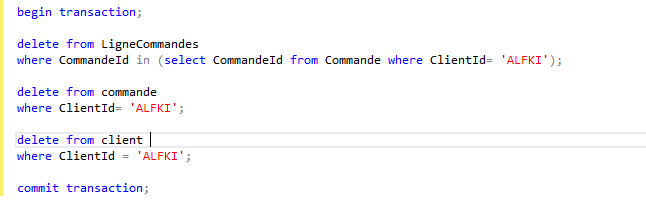
**Exemple :**

On prend l’exemple de la suppression DELETE :

Pour supprimer le client il fallait passer par la suppression ses dépendance dans la table commande et a son tours dans la table lignecommande

Alors on crée une TRANSACTION avec BEGIN TRANSACTION et fini avec un COMMIT TRANSACTION ou un ROLLBACK TRANSACTION, on regroupe des requêtes pour faire un seule block.

Cela permet de s’assurer que la requête de la transaction a été exécutée complètement ou pas du tout et correctement



# La gestion des erreurs :

Pour savoir si on doit annuler une transaction ou pas on doit s’assurer que la transaction a été exécuter correctement et pour faire il faut utiliser « un bloc try catch » ou on va créer un block avec la commande BEGIN TRY et BEGIN CATCH.

**La Syntaxe :**

# BEGIN TRY

{ sql\_statement | statement\_block }

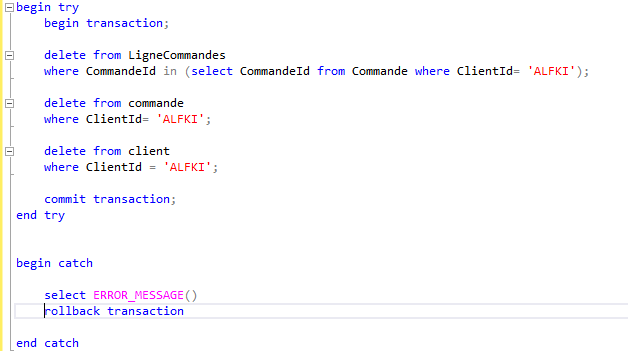
END TRY

# BEGIN CATCH

[ { sql\_statement | statement\_block } ]

END CATCH

**Exemple :**



Le BEGIN TRY : là on crée notre bloc à tester.

Le BEGIN CATCH : qui va récupérer l’exécution on cas d’erreur avec le SELECT ERROR\_X ou X est le type d’erreur qu’on veut récupérer.

Ensuite le ROLLBACK fait annuler la transaction et le code comme s’il n’a jamais été exécuter.

Dans notre exemple on a utilisé ERROR\_MESSAGE  qui renvoie le texte complet du message d’erreur. Le texte comprend les valeurs fournies pour tous les paramètres remplaçables, tels que les longueurs, les noms d'objet ou les heures.

# TRUNCATE TABLE :

En SQL, la commande TRUNCATE permet de supprimer toutes les données d’une table sans supprimer la table en elle-même

**La Syntaxe :**

TRUNCATE TABLE `nom\_table`

**DROP :**

Permet de supprimer une table dans SQL et ses données.

**La syntaxe :**

DROP TABLE nom\_table

La différence entre DROP et TRUNCATE :

La commande DROP a pour but de supprimer les données ainsi que la table qui les contient, et elle est semblable à l’instruction DELETE sans utilisation de WHERE. Tandis que La commande TRUNCATE permet de supprimer toutes les données d’une table sans supprimer la table en elle-même.

----------------------TD-------------------

# TD1 -FILM

1) Création de la table FILM

CREATE TABLE FILM

(

Id\_FILM INT,

Id\_Realisateur INT,

Titre VARCHAR(50),

Genre VARCHAR(15),

Année INT,

CONSTRAINT PK\_FILM primary key (Id\_FILM)

);

2)Afficher le contenu de la table PERSONNE

3--CREATE TABLE PERSONNE

(

Id\_Pers INT,

Nom VARCHAR(10),

Prénom VARCHAR(10),

CONSTRAINT PK\_PERSONNE primary key (Id\_Pers)

);

-- 4) Selectionner dans la table FILM les films qui sont du genre Drame et Policier

INSERT INTO FILM (Id\_FILM, Id\_Realisateur, Titre, Genre, Année)

VALUES

(01, 15, 'Crash', 'Drame', 1996),

(02, 15, 'Faux-Semblants', 'Epouvante', 1988),

(03, 14, 'Pulp Fiction', 'Policier', 1994),

(04, 13, 'Breaking the waves', 'Drame', 1996),

(05, 13, 'Dogville', 'Drame', 2002),

(06, 12, 'Alamo', 'Western', 1960),

(07, 18, 'Dangereusement vôtre', 'Espionnage', 1985),

(08, 19, 'Chasseur blanc, coeur noir', 'Drame', 1989),

(10, 21, 'American Beanly', 'Drame', 1999);

5- on insert les données de la table PERSONNE\*/

INSERT INTO PERSONNE(Id\_Pers, Nom, Prénom)

VALUES

(01, 'Kidman', 'Nicole'),

(02,'Bettany', 'Paul'),

(03, 'Watson', 'Emily'),

(04, 'Skarsgard', 'Stellan'),

(05, 'Travolta', 'John'),

(06, 'L. Jackson', 'Samuel'),

(07, 'Willis', 'Bruce'),

(08, 'Irons', 'Jeremy'),

(09, 'Spader', 'James'),

(10, 'Hunter', 'Holly'),

(11, 'Arquette', 'Rosanna'),

(12, 'Wayne', 'John'),

(13, 'Von Trier', 'Lars'),

(14, 'Tarantino', 'Quentin'),

(15, 'Cronenberg', 'David'),

(16, 'Mazursky', 'Paul'),

(17, 'Jones', 'Grace'),

(18, 'Glen', 'John'),

(19, 'Eastwood', 'Clint'),

(20, 'Spacey', 'Kevin'),

(21, 'Mendes', 'Sam'),

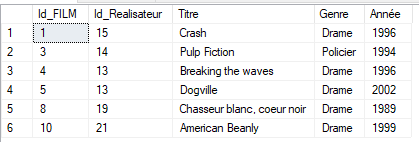
(22, 'Jolie', 'Angelina');

6-selection de genre Drame et policier

SELECT\*

FROM FILM

WHERE genre = 'Drame' or genre = 'Policier';



7-Supprimer les enregistrements de la table FILM datant avant 1995

DELETE FROM FILM

WHERE Année < 1995;

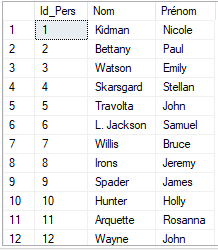
8--Supprimer la table FILM

DROP TABLE FILM;

9--Affichage du contenu de la table PERSONNE\*/

SELECT\*

FROM PERSONNE;



12--Ajouter ce nouvel enregistrement dans la table FILM

/\* 12 22 'L'age de glace' 'Animation' 2011 \*/

1. En utilisant un INSERT

INSERT INTO FILM

VALUES (12, 22, 'L''age de glace 4', 'Animation', 2011);



1. Ou inserer à partir d’un chemin

BULK INSERT FILM

FROM 'C:\Users\PERSO\Documents\ligne1.txt' /\*chemin à parcourir\*/

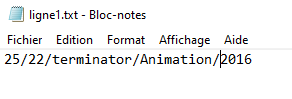
WITH

(format='CSV',// le formart du fichier texte

FIELDTERMINATOR = '/', // le separateur dans le fichier ou on importe

ROWTERMINATOR = '\n' // saut de ligne

);



# TD2-PUF

1-- creation de base de données

CREATE DATABASE PUF;

2--création des tables

CREATE TABLE USINE

(

NumU INT NOT NULL,

NomU VARCHAR(10),

VilleU VARCHAR(10),

CONSTRAINT PK\_USINE primary key (NumU)

);

3-insertion des données de la table USINE

INSERT INTO USINE (NumU, NomU, VilleU) VALUES (1,'dassault','Toulouse');

INSERT INTO USINE (NumU, NomU, VilleU) VALUES (2,'Wurth','Lyon');

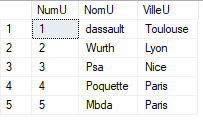
INSERT INTO USINE (NumU, NomU, VilleU) VALUES (3,'Psa','Nice');

INSERT INTO USINE (NumU, NomU, VilleU) VALUES (4,'Poquette','Paris');

INSERT INTO USINE (NumU, NomU, VilleU) VALUES (5,'Mbda','Paris');

4--SELECT\*

FROM USINE;



5--CREATE TABLE PRODUIT

(

NumP INT NOT NULL,

NomP VARCHAR(10),

Couleur VARCHAR(10),

Poids VARCHAR(10),

CONSTRAINT PK\_PRODUIT primary key (NumP)

);

6--insertion des données de la table PRODUIT

INSERT INTO PRODUIT (NumP, NomP, Couleur, Poids) VALUES (85,'lait','blanc',12),

(100,'café','noir',41),

(345,'baguette','marron',54),

(478,'fromage','jaune',12),

(521,'confiture','orange',41),

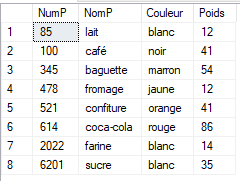
(614,'coca-cola','rouge',86),

(2022,'farine','blanc',14),

(6201,'sucre','blanc',35);

7--SELECT\*

FROM PRODUIT;



8--CREATE TABLE FOURNISSEUR

(

NumF INT NOT NULL,

NomF VARCHAR(20),

Statut VARCHAR(20),

VilleF VARCHAR(20),

CONSTRAINT PK\_FOURNISSEUR primary key (NumF)

);

9-- insertion des données de la table FOURNISSEUR

INSERT INTO FOURNISSEUR (NumF, NomF, Statut, VilleF) VALUES (2,'Agidra','Statut1','Lyon'),

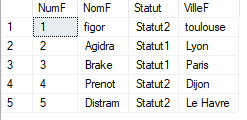
(3,'Brake','Statut1','Paris'),

(4,'Prenot','Statut2','Dijon'),

(5,'Distram','Statut2','Le Havre');

10--SELECT\*

FROM FOURNISSEUR;



11--CREATE TABLE Puf

(

NumP INT NOT NULL,

NumU INT NOT NULL,

NumF INT NOT NULL,

Quantité INT,

CONSTRAINT FK\_USINE FOREIGN KEY (NumU) REFERENCES USINE(NumU),

CONSTRAINT FK\_PRODUIT FOREIGN KEY (NumP) REFERENCES PRODUIT(NumP),

FOREIGN KEY (NumF) REFERENCES FOURNISSEUR(NumF)

);

12-insertion des données de la table PUF

INSERT INTO PUF (NumP, NumU, NumF, Quantité) VALUES (614,2,3,125),

(100,2,3,856),

(100,2,1,452),

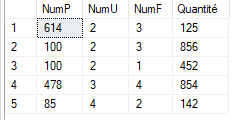
(478,3,4,854),

(85,4,2,142);

13—

SELECT\*

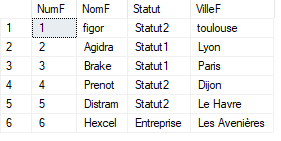
FROM PUF;



14--Ajout d'un nouveau fournisseur

INSERT INTO FOURNISSEUR (NumF, NomF, Statut, VilleF)

VALUES (6, 'Hexcel', 'Entreprise', 'Les Avenières');



15--Suppression de tous les produits de couleur noire et de numéro compris entre 100 et 1999

a)

DELETE

FROM PUF

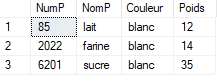
WHERE NumP BETWEEN 100 AND 1999;

b)

DELETE

FROM PRODUIT

WHERE Couleur = 'noire' OR NumP BETWEEN 100 AND 1999;



17--Changer la ville du fournisseur 3 par Toulouse

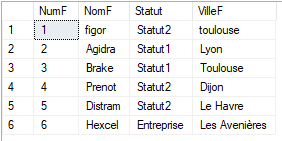
UPDATE FOURNISSEUR

SET VilleF = 'Toulouse'

WHERE NumF = 3;

18--SELECT\*

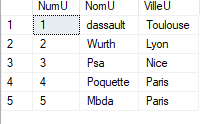
FROM FOURNISSEUR;



19-- Donner le numéro, le nom et la ville de toutes les usines

SELECT NumU, NomU, VilleU

FROM USINE;

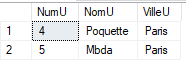


20--Donner le numéro, le nom et la ville de toutes les usines de Paris

SELECT NumU, NomU, VilleU

FROM USINE

WHERE VilleU = 'Paris';



21--Donner les numéros des fournisseurs qui approvisionnent l’usine de numéro 2 en produit de numéro

--100

SELECT NumF

FROM PUF

WHERE NumU = 2 AND NumP = 100;



22--Donner les noms et les couleurs des produits livrés par le fournisseur de numéro 2

SELECT NomP, Couleur

FROM PRODUIT

INNER JOIN PUF ON PUF.NumP = PRODUIT.NumP

WHERE NumF = 2;



23--Donner les numéros des fournisseurs qui approvisionnent l’usine de numéro 2 en un produit rouge

SELECT NumF

FROM PUF

INNER JOIN PRODUIT ON PUF.NumP = PRODUIT.NumP

WHERE NumU = 2 AND Couleur = 'rouge';



24--Donner les noms des fournisseurs qui approvisionnent une usine de Paris ou de Créteil en produit rouge

SELECT NomF

From PUF

INNER JOIN PRODUIT ON PUF.NumP = PRODUIT.NumP

INNER JOIN FOURNISSEUR ON PUF.NumF = FOURNISSEUR.NumF

INNER JOIN USINE ON PUF.NumU = USINE.NumU

WHERE VilleU = 'Paris' OR VilleU = 'Créteil' AND Couleur = 'rouge';



25--Donner les numéros des produits livrés à une usine par un fournisseur de la même ville

select nump

from USINE

join PUF on PUF.NumU = USINE.NumU

join FOURNISSEUR on FOURNISSEUR.NumF=puf.NumF

WHERE VilleU = VilleF;

26--Donner les numéros des produits livrés à une usine de Paris par un fournisseur de Paris

SELECT NumP

FROM PUF

INNER JOIN USINE ON PUF.NumU = USINE.NumU

INNER JOIN FOURNISSEUR ON PUF.NumF = FOURNISSEUR.NumF

WHERE VilleU = 'Paris' AND VilleF = 'Paris';

27--Donner les numéros des usines qui ont au moins un fournisseur qui n’est pas de la même ville

SELECT NumP

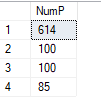
FROM PUF

INNER JOIN USINE ON PUF.NumU = USINE.NumU

INNER JOIN FOURNISSEUR ON PUF.NumF = FOURNISSEUR.NumF

--WHERE VilleU <> VilleF;

WHERE VilleU IN (SELECT VilleF FROM FOURNISSEUR);



28/\* 2nd methode (pour recuperer les noms des villes)\*/

SELECT VilleU

FROM PUF,USINE, FOURNISSEUR

where PUF.NumU = USINE.NumU and PUF.NumF = FOURNISSEUR.NumF

EXCEPT (

SELECT VilleF

FROM PUF, USINE, FOURNISSEUR

where PUF.NumU = USINE.NumU and PUF.NumF = FOURNISSEUR.NumF and VilleU <> VilleF

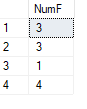
);

29--Donner les numéros des fournisseurs qui approvisionnent à la fois des usines de numéros 2 et 3

SELECT NumF

FROM PUF

WHERE NumU = 2 OR NumU = 3;



30--Donner les numéros des usines qui utilisent au moins un produit disponible chez le fournisseur de numéro 3

SELECT NumU

FROM PUF

WHERE NumP IN (SELECT NumP FROM PUF WHERE NumF = 3);

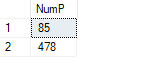


31--Donner le numéro du produit le plus léger (les numéros si plusieurs produits ont ce même poids)

SELECT NumP

FROM PRODUIT

WHERE Poids IN (SELECT MIN(DISTINCT Poids) FROM PRODUIT);



32--Donner le numéro des usines qui ne reçoivent aucun produit rouge d’un fournisseur parisien

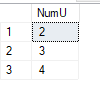
SELECT DISTINCT NumU

FROM PUF

INNER JOIN PRODUIT ON PUF.NumP = PRODUIT.NumP

INNER JOIN FOURNISSEUR ON PUF.NumF = FOURNISSEUR.NumF

WHERE Couleur != 'rouge' AND VilleF != 'Paris';



33--Donner les numéros des fournisseurs qui fournissent au moins un produit fourni par au moins un fournisseur qui fournit au moins un produit rouge

SELECT DISTINCT NumF

FROM PUF

WHERE NumP IN (SELECT NumP

FROM PUF

WHERE NumF IN ( SELECT NumF

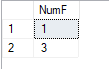
FROM PUF

WHERE NumP IN ( SELECT NumP

FROM PRODUIT

WHERE Couleur = 'rouge'

)));



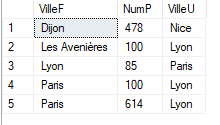
34--Donner tous les triplets (VilleF, NumP, VilleU) tels qu’un fournisseur de la première ville VilleF approvisionne une usine de la deuxième ville VilleU avec un produit NumP

SELECT DISTINCT VilleF, NumP, VilleU

FROM PUF

JOIN FOURNISSEUR ON PUF.NumF = FOURNISSEUR.NumF

JOIN USINE ON PUF.NumU = USINE.NumU;



35--Même question que précédemment mais sans les triplets où les deux villes sont identiques

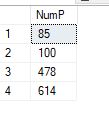
SELECT DISTINCT VilleF, NumP, VilleU

FROM PUF

INNER JOIN FOURNISSEUR ON PUF.NumF = FOURNISSEUR.NumF

INNER JOIN USINE ON PUF.NumU = USINE.NumU

WHERE VilleF = VilleU;



36--Donner les numéros des produits qui sont livrés à toutes les usines de Paris

SELECT DISTINCT NumP

FROM PUF

INNER JOIN USINE ON PUF.NumU = USINE.NumU

WHERE VilleU = 'Paris';

37--Donner les numéros des fournisseurs qui approvisionnent toutes les usines avec un même produit

SELECT NumF

FROM PUF

WHERE NumP IN (SELECT NumP

FROM PUF

WHERE NumU IN (SELECT NumU

FROM PUF ));

39--Donner les numéros des usines qui achètent au fournisseur de numéro 3 tous les produits qu’il

fournit

SELECT NumU

FROM PUF

WHERE NumP IN (

SELECT NumP

FROM PUF

WHERE NumF IN (

SELECT NumF

FROM PUF

WHERE NumF = 3

));

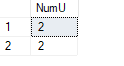


40--Donner les numéros des usines qui s’approvisionnent uniquement chez le fournisseur de numéro 3

SELECT NumU

FROM PUF

WHERE NumF = 3;



41--Supprimer tous les produits de couleur noire et de numéros compris entre 100 et 1999

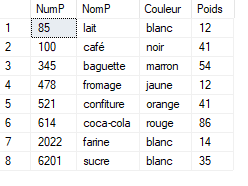
DELETE

FROM PRODUIT

WHERE Couleur = 'noire' AND NumP BETWEEN 100 AND 1999;

SELECT \*

FROM PRODUIT



# TP- Gavasoft

1--Creation de la base de données

CREATE DATABASE Gavasoft;

Création des tables

2--Table Dept

CREATE TABLE Dept (

NumD INT NOT NULL, NomD VARCHAR(10), Lieu VARCHAR(10),

CONSTRAINT PK\_Dept primary key (NumD)

);

3—

SELECT\*

FROM Dept;

4--Insertion des données de la table Dept

INSERT INTO Dept (NumD, NomD, Lieu)

VALUES (1,'Droit','Créteil'),

(2,'Commerce','Boston');

SELECT\*

FROM Dept;



6--table Emp

CREATE TABLE Emp (

NumE INT NOT NULL, NomE VARCHAR(20),

Fonction VARCHAR(20), NumS INT,

Embauche DATE, Salaire INT NOT NULL, Comm INT,

NumD INT,

CONSTRAINT PK\_Emp primary key (NumE),

CONSTRAINT FK\_Dept FOREIGN KEY (NumD) REFERENCES Dept(NumD)

);

7--Insertion des données de la table Emp

INSERT INTO Emp (NumE, NomE, Fonction, NumS, Embauche, Salaire, Comm, NumD) VALUES

(1, 'Grava', 'Président', NULL, '10/10/1979', 10000, NULL, NULL), (2, 'Guimezanes', 'Doyen', 1, '01/10/2006', 5000, NULL, 1),

(3, 'Toto', 'Stagiare', 1, '01/10/2006', 0, NULL, 1),

(4, 'Al-Capone', 'Commercial', 2, '01/10/2006', 5000, 100, 2),

(5, 'Patrick', 'Commercial', 3, '03-11-2006', 6000, 200, 1);

8 SELECT\* FROM Emp;

--1 Donner la liste des employés ayant une commission (non NULL) classé par commission décroissante

SELECT NomE, Comm FROM Emp

WHERE Comm IS NOT NULL ORDER BY Comm

DESC;



--2 Donner les noms des personnes embauchées depuis le 01-09-2006

SELECT NomE, Embauche

FROM Emp

WHERE Embauche >= '01-09-2006';

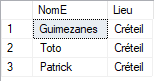


--3Donner la liste des employés travaillant à Créteil

SELECT NomE, Lieu

FROM Emp

INNER JOIN Dept ON Emp.NumD = Dept.NumD WHERE Lieu = 'Créteil';



--4Donner la liste des subordonnés de "Guimezanes"

SELECT NomE, Salaire

FROM Emp

WHERE Salaire > 5000;



--5Donner la moyenne des salaires

SELECT AVG(Salaire) as moyenne

FROM Emp;



--6Donner le nombre de commissions non NULL

SELECT COUNT(Comm) nbr\_comission

FROM Emp

WHERE Comm IS NOT NULL;



--7Donner la liste des employés gagnant plus que la moyenne des salaires

de l’entreprise

SELECT NomE, Salaire

FROM Emp

WHERE Salaire > (SELECT AVG(Salaire)

FROM Emp

); 

15---

# TP-BD Banque

1--Création de la BD Banque

CREATE DATABASE Banque;

Création des tables

2--Création de tables AGENCE

CREATE TABLE AGENCE (

Num\_Agence INT NOT NULL,

Nom VARCHAR(50),

Ville VARCHAR(50),

Actif INT NOT NULL

CONSTRAINT PK\_AGENCE PRIMARY KEY (Num\_Agence)

);

3--Insertion des données de la table AGENCE

INSERT INTO AGENCE (Num\_Agence, Nom, Ville, Actif) VALUES ('1','Agence1','paris',2000),

('2','Agence2','Paris- Etoile',2025),

('3','Paris-Bastille','La Rochelle',4158),

('4','Agence4','Orsay',4158),

('5','Agence5','nancy',2552),

('6','Agence6','Orsay',3052),

('7','Agence7','Paris- Rambuteau',3052),

SELECT\*

FROM AGENCE;



5--création de tables CLIENT

CREATE TABLE CLIENT (

Num\_Client INT NOT NULL,

Nom VARCHAR(50),

Ville VARCHAR(50)

CONSTRAINT PK\_CLIENT PRIMARY KEY (Num\_Client)

);

6/\* insertion des données de la table CLIENT\*/

INSERT INTO CLIENT (Num\_Client, Nom, Ville) VALUES ('1','Céline Grignon','toulouse'); INSERT INTO CLIENT (Num\_Client, Nom, Ville) VALUES ('2','Dupont','paris');

INSERT INTO CLIENT (Num\_Client, Nom, Ville) VALUES ('3','Moore Lavoie','paris'); INSERT INTO CLIENT (Num\_Client, Nom, Ville) VALUES ('4','Claude','grenoble');

INSERT INTO CLIENT (Num\_Client, Nom, Ville) VALUES ('5','Adélaïde Dupont','dijon');

INSERT INTO CLIENT (Num\_Client, Nom, Ville) VALUES ('6','Jacquenett Plouffe','marseille');

INSERT INTO CLIENT (Num\_Client, Nom, Ville) VALUES ('7','Éléonore Levasseur','le havre');

SELECT\*

FROM CLIENT;



8--création de tables COMPTE

CREATE TABLE COMPTE (

Num\_Compte INT NOT NULL,

Num\_Agence INT NOT NULL,

Num\_Client INT NOT NULL,

Solde INT

CONSTRAINT PK\_COMPTE PRIMARY KEY (Num\_Compte),

CONSTRAINT FK\_AGENCE FOREIGN KEY (Num\_Agence) REFERENCES AGENCE(Num\_Agence),

CONSTRAINT FK\_CLIENT FOREIGN KEY (Num\_Client) REFERENCES CLIENT(Num\_Client)

);

9/\* insertion des données de la table COMPTE\*/

INSERT INTO COMPTE (Num\_Compte,Num\_Agence,Num\_Client, Solde)

VALUES (1,2,4,22567),

(2,3,4,5864),

(3,1,5,84565),

(4,2,1,6212),

(5,3,2,56582),

(6,4,3,15623),

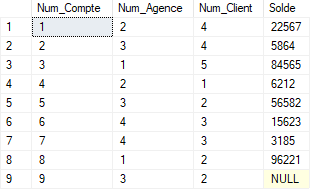
(7,4,3,3185),

(8,1,2,96221),

(9,3,2,null),

SELECT\*

FROM COMPTE;



11-Création de tables COMPTE

CREATE TABLE EMPRUNT (

Num\_Emprunt INT NOT NULL,

Num\_Agence INT NOT NULL,

Num\_Client INT NOT NULL,

Montant INT NOT NULL

CONSTRAINT PK\_EMPRUNT PRIMARY KEY (Num\_Emprunt),

CONSTRAINT FK\_EMPRUNT\_AGENCE FOREIGN KEY (Num\_Agence) REFERENCES AGENCE(Num\_Agence),

CONSTRAINT FK\_EMPRUNT\_CLIENT FOREIGN KEY (Num\_Client) REFERENCES CLIENT(Num\_Client)

);

12/\* insertion des données de la table EMPRUNT\*/

INSERT INTO EMPRUNT (Num\_Emprunt, Num\_Agence, Num\_Client, Montant)

VALUES (1,2,5,85624),

(2,2,5,85624),

(3,1,1,85624),

(4,4,2,85624),

(5,4,1,85624),

(6,4,3,85624),

(7,3,4,85624),

(8,7,3,85624),

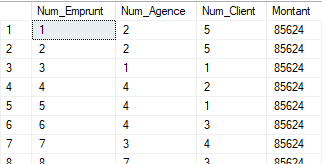
(9,7,4,85624),

(10,7,2,85624);

13-

SELECT\*

FROM EMPRUNT;



14-- Liste des agences ayant des comptes-clients

SELECT DISTINCT Nom

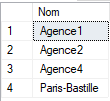
FROM AGENCE

INNER JOIN COMPTE ON AGENCE.Num\_Agence = COMPTE.Num\_Agence

WHERE COMPTE.Num\_AGENCE in (

SELECT Num\_Agence FROM AGENCE

);



15--Clients ayant un compte à La Rochelle

SELECT DISTINCT CLIENT.Nom FROM COMPTE

INNER JOIN AGENCE ON AGENCE.Num\_Agence = COMPTE.Num\_Agence

INNER JOIN CLIENT ON COMPTE.Num\_Client = CLIENT.Num\_Client

WHERE AGENCE.Ville = 'La Rochelle';



16-- Clients ayant un compte ou un emprunt à La Rochelle

(SELECT DISTINCT CLIENT.Nom

FROM COMPTE

INNER JOIN AGENCE ON AGENCE.Num\_Agence = COMPTE.Num\_Agence

INNER JOIN CLIENT ON COMPTE.Num\_Client = CLIENT.Num\_Client

WHERE AGENCE.Ville = 'La Rochelle')

UNION all

(SELECT DISTINCT CLIENT.Nom FROM EMPRUNT

INNER JOIN AGENCE ON AGENCE.Num\_Agence = EMPRUNT.Num\_Agence

INNER JOIN CLIENT ON CLIENT.Num\_Client = EMPRUNT.Num\_Client

WHERE AGENCE.Ville = 'La Rochelle')



17-- Clients ayant un compte et un emprunt à La Rochelle

(SELECT DISTINCT CLIENT.Nom

FROM COMPTE

INNER JOIN AGENCE ON AGENCE.Num\_Agence = COMPTE.Num\_Agence

INNER JOIN CLIENT ON COMPTE.Num\_Client = CLIENT.Num\_Client

WHERE AGENCE.Ville = 'La Rochelle')

INTERSECT

(SELECT DISTINCT CLIENT.Nom FROM EMPRUNT

INNER JOIN AGENCE ON AGENCE.Num\_Agence = EMPRUNT.Num\_Agence

INNER JOIN CLIENT ON CLIENT.Num\_Client = EMPRUNT.Num\_Client

WHERE AGENCE.Ville = 'La Rochelle');



18-- Clients ayant un compte et pas d’emprunt à La Rochelle

(SELECT DISTINCT CLIENT.Nom

FROM COMPTE

INNER JOIN AGENCE ON AGENCE.Num\_Agence = COMPTE.Num\_Agence

INNER JOIN CLIENT ON COMPTE.Num\_Client = CLIENT.Num\_Client

WHERE AGENCE.Ville = 'La Rochelle')

EXCEPT

(SELECT DISTINCT CLIENT.Nom

FROM EMPRUNT

INNER JOIN AGENCE ON AGENCE.Num\_Agence = EMPRUNT.Num\_Agence

INNER JOIN CLIENT ON CLIENT.Num\_Client = EMPRUNT.Num\_Client

WHERE AGENCE.Ville = 'La Rochelle');

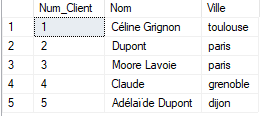


19-- Clients ayant un compte et nom de la ville où ils habitent

SELECT DISTINCT COMPTE.Num\_Client, CLIENT.Nom, CLIENT.Ville

FROM COMPTE

INNER JOIN CLIENT ON CLIENT.Num\_Client = COMPTE.Num\_Client;



20-- Clients ayant un compte à Paris-Etoile et nom de la ville où ils habitent

SELECT CLIENT.Nom, CLIENT.Ville

FROM COMPTE

INNER JOIN CLIENT ON CLIENT.Num\_Client = COMPTE. Num\_Client

INNER JOIN AGENCE ON AGENCE.Num\_Agence = COMPTE.Num\_Agence

WHERE AGENCE.Nom = 'Paris-Etoile';

-- 21 les Clients ayant un compte dans une agence où Claude a un compte

SELECT Nom

FROM CLIENT

WHERE Num\_Client in (

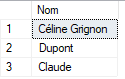
SELECT Num\_Client

FROM COMPTE

WHERE Num\_Agence in (SELECT Num\_Agence

FROM CLIENT cl, COMPTE co

WHERE CL.Num\_Client = CO.Num\_Client and Nom = 'Claude'));



-- 22 les Agences ayant un actif moin élevé que toute agence d'Orsay

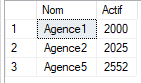
SELECT Nom , Actif

FROM AGENCE

WHERE Actif < all ( SELECT Actif

FROM AGENCE

WHERE Ville = 'Orsay');



-- 23 Clients ayant un compte dans chaque agence d'Orsay

SELECT DISTINCT cl.Nom, ag.Ville

FROM CLIENT cl

JOIN COMPTE co ON co.Num\_Client = cl.Num\_Client

JOIN AGENCE Ag ON co.Num\_Agence = ag.Num\_Agence

WHERE ag.Ville = 'Orsay';



24-- les Clients ayant un compte dans au-moins une agence d'Orsay

SELECT DISTINCT Cl.Nom, ag.Ville

FROM CLIENT cl, COMPTE co, AGENCE ag

WHERE cl.Num\_Client = co.Num\_Client AND co.Num\_Agence = Ag.Num\_Agence AND Ag.Ville = 'Orsay';

25--le nom des Emprunteurs de l'agence Paris-Rambuteau classés par ordre alphabétique

SELECT distinct cl.Nom

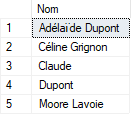
FROM CLIENT cl

JOIN EMPRUNT em ON em.Num\_Client=cl.Num\_Client

JOIN AGENCE Ag ON em.Num\_Agence = ag.Num\_Agence

where ag.Ville like'%Paris%'

order by cl.Nom



26-- Solde moyen des comptes-clients des agences dont le solde moyen est > 10 000

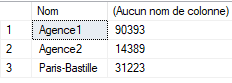
SELECT Nom, AVG(Solde)

FROM COMPTE Co

INNER JOIN AGENCE Ag ON AG.Num\_Agence = CO.Num\_Agence

GROUP BY Nom

HAVING AVG(Solde) > 10000;



27-- Nombre de clients habitant Paris

SELECT COUNT(Num\_Client)

FROM CLIENT

WHERE Ville = 'Paris';



28-- Nombre de clients de l'agence “Paris-Bastille” n'ayant pas leur adresse dans la relation CLIENT

SELECT COUNT(Num\_Client)

FROM CLIENT

WHERE Ville = NULL AND Num\_Client IN

(select cl.Num\_Client

from CLIENT cl

left join COMPTE co on cl.Num\_Client= co.Num\_Client

left join AGENCE ag on ag.Num\_Agence=co.Num\_Agence

WHERE ag.Nom = 'Paris-Bastille')

-- 29 Diminuer l'emprunt de tous les clients habitant Marseille de 5%

UPDATE EMPRUNT SET Montant = Montant\*0.95

WHERE Num\_Client IN (

SELECT Num\_Client FROM CLIENT

WHERE Ville = 'Marseille'

);

30-- Fermer les comptes de Dupont

DELETE FROM COMPTE

WHERE Num\_Client in (

SELECT Num\_Client

FROM CLIENT

WHERE Nom = 'Dupont'

);

31-- Supprimer les agences qui n’ont pas de compte

begin try

begin transaction

delete from EMPRUNT

where Num\_Agence in (

select ag.Num\_Agence FROM AGENCE ag

WHERE Num\_Agence NOT IN (

SELECT distinct ag.Num\_Agence

FROM COMPTE co

INNER JOIN AGENCE ag on co.Num\_Agence = ag.Num\_Agence )

);

DELETE FROM AGENCE

WHERE Num\_Agence NOT IN (

SELECT distinct ag.Num\_Agence

FROM COMPTE co

INNER JOIN AGENCE ag on co.Num\_Agence = ag.Num\_Agence )

commit transaction

end try

BEGIN CATCH

select ERROR\_MESSAGE()

select ERROR\_NUMBER()

rollback transaction

END CATCH

34--20 Les actifs toute agence d'Orsay

SELECT Actif

FROM AGENCE

WHERE Ville = 'Orsay'



34--

# TP-VOITURE

1Création de la base de données CAR

CREATE DATABASE vehicule;

2/\*création des tables\*/

CREATE TABLE VOITURE (

numserie VARCHAR(50),

couleur VARCHAR(10),

prix INT NOT NULL,

cout INT NOT NULL,

marque VARCHAR(20),

modele VARCHAR(20)

CONSTRAINT PK\_VOITURE PRIMARY KEY (numserie)

);

3/\* insertion des données de la table AGENCE\*/

INSERT INTO VOITURE(numserie, couleur, prix, cout, marque, modele)

VALUES ('M10BMWVP015K188', 'gris', 35000, 40000, 'BMW', '430d Coup');

INSERT INTO VOITURE(numserie, couleur, prix, cout, marque, modele)

VALUES ('M10FERVP000P145', 'rouge', 1600000, 1730000, 'FERRARI', 'LaFerrari');

INSERT INTO VOITURE(numserie, couleur, prix, cout, marque, modele)

VALUES ('MKA1002RR404', 'gris', 9521, 9652, 'KIA', 'PICANTO');

INSERT INTO VOITURE(numserie, couleur, prix, cout, marque, modele)

VALUES ('MLA2901BJ002', 'orange', 604085, 605385, 'LAMBORGHINI', 'DIABLO');

INSERT INTO VOITURE(numserie, couleur, prix, cout, marque, modele)

VALUES ('M10DACVP000Z467', 'gris', 1200, 1652, 'DACIA', 'LODGY');

SELECT\*

FROM VOITURE;

4\* insertion des données de la table ARRIVEE

CREATE TABLE ARRIVEE (

numserie VARCHAR(50),

date\_A DATE,

magasin VARCHAR(20)

CONSTRAINT PK\_ARRIVEE PRIMARY KEY (numserie, date\_A),

CONSTRAINT FK\_ARRIVEE FOREIGN KEY (numserie) REFERENCES VOITURE(numserie)

);

5/\* insertion des données de la table ARRIVEE\*/

INSERT INTO ARRIVEE (numserie, date\_A, magasin)

VALUES ('M10BMWVP015K188','05-08-2017','magasin1');

INSERT INTO ARRIVEE (numserie, date\_A, magasin)

VALUES ('M10FERVP000P145','06-08-2017','magasin2');

INSERT INTO ARRIVEE (numserie, date\_A, magasin)

VALUES ('MKA1002RR404','07-08-2017','magasin3');

INSERT INTO ARRIVEE (numserie, date\_A, magasin)

VALUES ('MLA2901BJ002','08-08-2017','magasin4');

SELECT\*

FROM ARRIVEE;

CREATE TABLE CLIENT(

nom VARCHAR(20) UNIQUE,

prenom VARCHAR(20) UNIQUE,

adresse VARCHAR(20)

CONSTRAINT PK\_CLIENT PRIMARY KEY (nom, prenom)

);

6/\* insertion des données de la table CLIENT\*/

INSERT INTO CLIENT (nom, prenom, adresse)

VALUES ('Benjamin', 'Coralie', 'adresse1');

INSERT INTO CLIENT (nom, prenom, adresse)

VALUES ('Marois', 'Amaury', 'adresse2');

INSERT INTO CLIENT (nom, prenom, adresse)

VALUES ('Quessy', 'Christiane', 'adresse3');

INSERT INTO CLIENT (nom, prenom, adresse)

VALUES ('Fournier', 'Jacquenett', 'adresse4');

INSERT INTO CLIENT (nom, prenom, adresse)

VALUES ('Marcoux', 'Bruce', 'adresse5');

SELECT\*

FROM CLIENT;

7--insertion des données de la table VENDEUR

CREATE TABLE VENDEUR(

nom VARCHAR(20) UNIQUE,

prenom VARCHAR(20) UNIQUE,

adresse VARCHAR(50),

salaire\_fixe INT NOT NULL,

magasin VARCHAR(20)

CONSTRAINT PK\_VENDEUR PRIMARY KEY (nom, prenom)

);

8--/\* insertion des données de la table VENDEUR\*/

INSERT INTO VENDEUR(nom, prenom, adresse, salaire\_fixe, magasin) VALUES ('Latourelle', 'Comforte', 'adresse1', 5300, 'magasin1');

INSERT INTO VENDEUR(nom, prenom, adresse, salaire\_fixe, magasin) VALUES ('Morneau', 'Evrard', 'adresse2', 3400, 'magasin2');

INSERT INTO VENDEUR(nom, prenom, adresse, salaire\_fixe, magasin) VALUES ('Desilets', 'Jeoffroi', 'adresse3', 6800, 'magasin3');

INSERT INTO VENDEUR(nom, prenom, adresse, salaire\_fixe, magasin) VALUES ('Coupart', 'Esperanza', 'adresse4', 2100, 'magasin4');

INSERT INTO VENDEUR(nom, prenom, adresse, salaire\_fixe, magasin) VALUES ('Parmentier', 'Felicien', 'adresse5', 1800, 'magasin5');

9--

SELECT\*

FROM VENDEUR;

10--insertion des données de la table VENTE

CREATE TABLE VENTE(

numero INT NOT NULL,

date\_V DATE,

prix\_achat INT NOT NULL,

numserie VARCHAR(50),

nomC VARCHAR(20),

prenomC VARCHAR(20),

nomv VARCHAR(20),

prenomv VARCHAR(20)

CONSTRAINT PK\_VENTE PRIMARY KEY (numero),

CONSTRAINT FK\_VENTE FOREIGN KEY (numserie) REFERENCES VOITURE(numserie),

CONSTRAINT FK\_VENDU FOREIGN KEY (nomc, prenomc) REFERENCES CLIENT(nom,prenom),

CONSTRAINT FK\_AVENDRE FOREIGN KEY (nomv, prenomv) REFERENCES VENDEUR(nom, prenom)

);

11/\* insertion des données de la table VENTE\*/

INSERT INTO VENTE(numero, date\_V, prix\_achat, numserie, nomc, prenomc, nomv, prenomv)

VALUES (1, '20051003', 1750000, 'M10FERVP000P145','Quessy', 'Christiane','Latourelle','Comforte');

INSERT INTO VENTE(numero, date\_V, prix\_achat, numserie, nomc, prenomc, nomv, prenomv)

VALUES (2, '20070415', 10590, 'MKA1002RR404','Marois', 'Amaury','Desilets', 'Jeoffroi');

INSERT INTO VENTE(numero, date\_V, prix\_achat, numserie, nomc, prenomc, nomv, prenomv)

VALUES (3, '20170416', 615367, 'MLA2901BJ002','Benjamin', 'Coralie','Morneau', 'Evrard');

INSERT INTO VENTE(numero, date\_V, prix\_achat, numserie, nomc, prenomc, nomv, prenomv)

VALUES (4, '20181006', 43442, 'M10BMWVP015K188','Marcoux', 'Bruce','Parmentier', 'Felicien');

SELECT\* FROM VENTE;

12--Donner la liste des voitures (numéro) vendues après le 15 avril 2007

SELECT numserie

FROM VENTE

WHERE date\_V > '20070415';

select \* from VOITURE

13--Donner la voiture qui rapporté le plus d’argent

SELECT marque, modele, VO.numserie

FROM VOITURE vo

join VENTE ve on VO.numserie = VE.numserie

GROUP BY marque, modele, VO.numserie

HAVING MAX(prix\_achat - cout) >= ALL (

SELECT MAX(prix\_achat - cout)

FROM VOITURE vo

join VENTE ve on VO.numserie = VE.numserie

);

14--Donner le vendeur ayant accordé le plus gros rabais

SELECT prenomv, nomv

FROM VENTE VE

INNER JOIN VOITURE VO ON VO.numserie = VE.numserie

WHERE prix - prix\_achat >= ALL (

SELECT (prix - prix\_achat) rabais

FROM VENTE

INNER JOIN VOITURE ON VOITURE.numserie = VENTE.numserie

) ;

15--Les bénéfices de chaque magasin pour le mois de janvier 2007

SELECT magasin, SUM(benefice) AS benefices

FROM (SELECT magasin, SUM((prix\_achat - cout -(prix - cout)/2) - salaire\_fixe) AS benefice

FROM VOITURE Vo

join VENTE Ve On vo.numserie=ve.numserie

join VENDEUR Ved on Ve.nomv = Ved.nom and Ve.prenomv = Ved.prenom

where date\_V BETWEEN '20070101' AND '20070630'

GROUP BY magasin) table\_benefice

GROUP BY magasin;

16--Le meilleur client (celui ayant rapporté le plus d’argent à l’entreprise)

SELECT nomc,prenomc

FROM VOITURE Vo

join VENTE Ve on VO.numserie = VE.numserie

GROUP BY nomc,prenomc

HAVING SUM(prix\_achat - cout - (prix - cout)/2) >= ALL (

SELECT SUM(prix\_achat - cout - 0.5\*(prix - cout))

FROM VOITURE, VENTE

WHERE VOITURE.numserie = VENTE.numserie

GROUP BY nomc,prenomc);

17--La marque pour laquelle on a accordé le plus de rabais

SELECT marque

FROM VOITURE

INNER JOIN VENTE ON VOITURE.numserie = VENTE.numserie

WHERE prix - prix\_achat >= ALL (

SELECT prix - prix\_achat

FROM VENTE

INNER JOIN VOITURE ON VOITURE.numserie = VENTE.numserie

);

18—modifier une donnée dans la table vendeur si elle existe ou inserrer la.

merge into vendeur ve

using (select 'maman' as nom,'cherie' as prenom ,'algerie'as adresse,'7500' as salaire\_fixe) source

on ve.nom=source.nom

when matched then

update set nom=source.nom

when not matched then

insert (nom,prenom,adresse,salaire\_fixe)

values (source.nom,source.prenom,source.adresse,source.salaire\_fixe);

19--Mettre en majuscule toute le nom de magasin

select upper(left(magasin,len(magasin)))

from VENDEUR

20 -- la première voiture vendue

select top 1 \*

from VOITURE vo

join VENTE ve on ve.numserie=vo.numserie

order by ve.date\_V desc

21par quel venderuet quand quel magasin ?

select top 1 vo.numserie,vo.modele,vo.marque,vd.nom,vd.prenom,vd.magasin

from VOITURE vo

join VENTE ve on ve.numserie=vo.numserie

join VENDEUR vd on vd.nom=ve.nomv

order by ve.date\_V desc

22toute les voiture grise et leur acheteur .

select marque,modele,couleur, nomC,prenomC

from VOITURE vo

left join VENTE ve on ve.numserie=vo.numserie

left join CLIENT cl on cl.nom=ve.nomC

where vo.couleur ='gris'

Ajouter une colonne carburant a la table voiture

alter table voiture

add carburant varchar(30)

Ajouter les types de carburant des voitures.

----------------------TP-------------------

# TP ORACLE – AIRBASE

1--création de table

create table Pilote (

NumP int primary key not null,

NomP varchar(30) ,

Adresse varchar(50),

Salaire int );

2--

create table Avion (

NumAv int primary key Not null,

NomAv varchar(30),

capacite int,

localisation varchar(30));

3--

create table vol (

NumV int primary key not null,

NumP Int not null,

NumAv int not null,

Dep\_T varchar(30),

Arr\_T varchar(30),

Dep\_H int ,

Arr\_H int,

FOREIGN KEY( NumP ) REFERENCES Pilote (NumP),

foreign key (NumAv) references Avion ( NumAv)

);

4--/\*insertion de données dans les table\*/

insert into avion (numav,nomav,capacite,localisation)

values (14,'boing250',250,'nice')

insert into avion (numav,nomav,capacite,localisation)

values (15,'airbus120',120,'marseille')

insert into avion (numav,nomav,capacite,localisation)

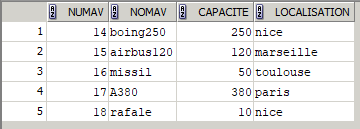
values ( 16, 'missil',50,'toulouse')

insert into avion (numav,nomav,capacite,localisation)

values (17,'A380',380,'paris')

insert into avion (numav,nomav,capacite,localisation)

values ( 18,'rafale',10,'nice')



5--

insert into Pilote(numP,nomp,adresse,salaire)

values (1,'yasmine','lyon',15000)

insert into Pilote(numP,nomp,adresse,salaire)

values(2,'samir','toulouse',17000)

insert into Pilote(numP,nomp,adresse,salaire)

values(3,'meriama','nice',13000)

insert into Pilote(numP,nomp,adresse,salaire)

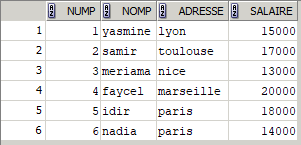
values(4,'faycel','marseille',20000)

insert into Pilote(numP,nomp,adresse,salaire)

values(5,'idir','paris',18000)

insert into Pilote(numP,nomp,adresse,salaire)

values(6,'nadia','paris',14000)



6--

insert into vol(numv,nump,numav,dep\_t,arr\_t,dep\_h,arr\_h)

values (4,1,15,'nice','toulous',18,19)

insert into vol(numv,nump,numav,dep\_t,arr\_t,dep\_h,arr\_h)

values (5,2,14,'lyon','paris',14,15)

insert into vol(numv,nump,numav,dep\_t,arr\_t,dep\_h,arr\_h)

values (7,3,17,'paris','marseille',9,11)

insert into vol(numv,nump,numav,dep\_t,arr\_t,dep\_h,arr\_h)

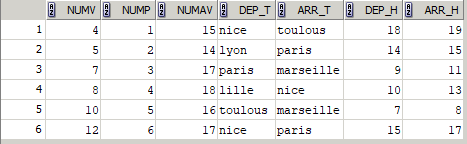
values (8,4,18,'lille','nice',10,13)

insert into vol(numv,nump,numav,dep\_t,arr\_t,dep\_h,arr\_h)

values (10,5,16,'toulous','marseille',7,8)

insert into vol(numv,nump,numav,dep\_t,arr\_t,dep\_h,arr\_h)

values (12,6,17,'nice','paris', 15,17)



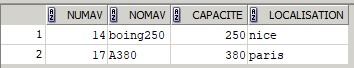
7--

/\*capacite d'avion superieur a 200\*/

select \*

from avion

where capacite >200

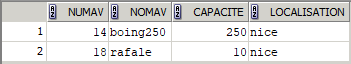


8/\*les avion localisé a nice\*/

select \*

from avion

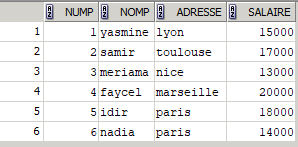
where localisation = 'nice'



9/\*afficher la table pilote\*/

Select \*

from pilote

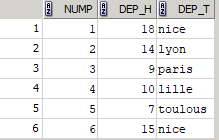


10/\*numéro de pilote /heure et lieu de départ de leur vol\*/

select pilote.Nump, vol.dep\_h, vol.dep\_t

from pilote

inner join vol on vol.nump =pilote.NumP



11/\*le nom, salaire et adresse des pilotes qui touche plus de 15000\*/

select nomP,Salaire,Adresse

from pilote

where salaire > 15000 and adresse ='paris'



12/\*nom et num d'avion qui ont une capacité moin de 350 \*/

select NumAv, nomAv

from avion

where localisation ='nice' and capacite < 350



13/\* les vols de Nice à Paris après 14h \*/

select \*

from vol

where dep\_t ='nice' and arr\_t = 'paris' and dep\_h >14



14/\*les pilote qui nont pas de vol\*/

select \*

from pilote

inner join vol on vol.nump= pilote.nump

where vol.nump is null

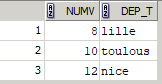


15/\*num et ville de depart des piloteID entre 4 et 10 \*/

select numv, dep\_t

from vol

where nump between 4 and 10

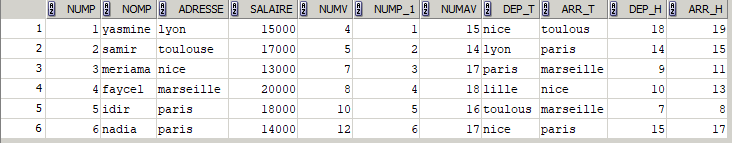


16/\*les pilotes et leur vol \*/

select \*

from pilote P

inner join vol V on p.nump= v.nump



# TP ORACLE HOTEL

1--

CREATE TABLE chambre (

Num\_Chambre INT PRIMARY KEY NOT NULL,

prix int,

Nbr\_Lit int ,

Nbr\_Pers int,

confort VARCHAR(30),

equ varchar(30)

);

2--select \* from chambre;

3—

create table client1 (

id\_Client int primary key not null ,

nom varchar(30),

prenom varchar(30),

adresse varchar(50));

4--select \* from client1;

5—

CREATE TABLE Reservation (

id\_Client INT NOT NULL,

Num\_Chambre INT NOT NULL,

Date\_ARV date,

Date\_DEP date,

FOREIGN KEY( id\_Client ) REFERENCES Client1 (id\_Client),

FOREIGN KEY( Num\_chambre ) REFERENCES Chambre (Num\_chambre)

);

6---lister la table reservation

select \* from reservation;

7---insertion des données dans la table chambre

insert into chambre (num\_chambre,prix,nbr\_lit,nbr\_pers,confort,equ)

values (10,80,01,02,'wc','non');

insert into chambre (num\_chambre,prix,nbr\_lit,nbr\_pers,confort,equ)

values (20,100,02,02,'douche','non');

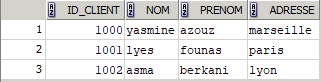
insert into chambre (num\_chambre,prix,nbr\_lit,nbr\_pers,confort,equ)

values (25,180,03,03,'bain','TV');

8--lister tous les client1

select \*

from client1



9---insertion des données dans la table client1

insert into client1 ( id\_client,nom,prenom,adresse)

values (1000,'yasmine','azouz','marseille');

insert into client1 ( id\_client,nom,prenom,adresse)

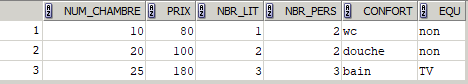
values(1001,'lyes','founas','paris');

insert into client1 ( id\_client,nom,prenom,adresse)

values(1002,'asma','berkani','lyon');

10---lister la table chambre

select \* from chambre;



11---insertion des données dans la table reservation

insert into Reservation(id\_client,num\_chambre,date\_arv,date\_dep)

values ( 1000,20,TO\_DATE('20140912','yyyymmdd'), TO\_DATE('20140921','yyyymmdd'))

insert into Reservation(id\_client,num\_chambre,date\_arv,date\_dep)

values ( 1001,10,TO\_DATE('20150630','yyyymmdd'), null)

12--

SELECT \*

from reservation



13--lister les chmabre qui ont une télévison

SELECT \*

from chambre

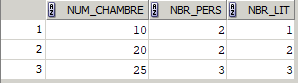
where equ = 'TV'



14--donner la capacite de chambre et de nombre de personne et nombre de chaque chambre

select num\_chambre ,nbr\_pers,nbr\_lit

from chambre



15--donner la capacite de l’hotel

select sum(nbr\_lit),sum(nbr\_pers),count(num\_chambre)

from chambre



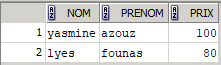
16--donner le prix par personne

select nom,prenom,prix

from chambre Ch

inner join reservation R on ch.num\_chambre= r.num\_chambre

inner join client1 Cl on cl.id\_client=r.id\_client



17---prix par personne des chambres avec télévision

select nom,prenom,prix

from chambre Ch

inner join reservation R on ch.num\_chambre= r.num\_chambre

inner join client1 Cl on cl.id\_client=r.id\_client

where ch.equ ='TV'

18--numero de chambre et client qui ont reservé pour le '12/09/2014'

select Ch.num\_chambre, cl.id\_client

from chambre Ch

inner join reservation R on ch.num\_chambre= r.num\_chambre

inner join client1 Cl on cl.id\_client=r.id\_client

where date\_arv = '12/09/2014'



19----les chambres qui coutent 80 ou plus

select \* --num\_chambre

from chambre

where prix <= 80



20--- nom, prénom, adresse des clients qui commences avec la lettre Y

select nom,prenom,adresse

from client1

where nom like 'y%'



21---le nombre de chambre dont le prix est entre 85 et 200 euro

select count(num\_chambre)

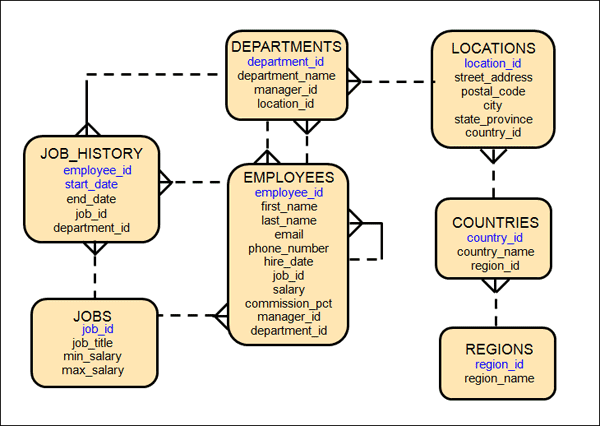
from chambre

where prix between 85 and 200



# TP ORACLE –HR

Le Schéma de la base de données.



select \*

from employees

where employee\_id in (

select \* from departments

where department\_name = 'Administration')

select \* from locations;

select \* from employees

where manager\_id = 102

where job\_id ='IT\_PROG';

select \* from jobs

select \* from job\_history

order by start\_date

select \* from regions

1--les employees qui sont manager

select distinct e1.employee\_id,e1.last\_name

from employees E1

join employees E2 on e1.employee\_id=e2.manager\_id

order by e1.employee\_id



2--les employees qui n’ont pas de manager

select \*

from employees

where manager\_id is null



3--les employees et leur contrat

select \*

from employees E

join job\_history JH on jh.employee\_id= e.employee\_id



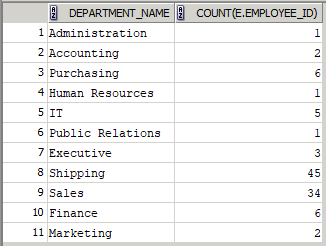
4--le nombre d'employe par departement

select d.department\_name , count (e.employee\_id)

from employees e

join departments D on e.department\_id=d.department\_id

group by d.department\_name



5--les employees les mieux payer dans chaque departement

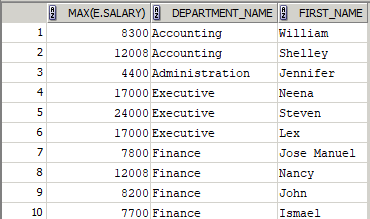
select max( e.salary),d.department\_name,e.first\_name

from employees e

join departments d on e.department\_id =d.department\_id

group by d.department\_name,e.first\_name

order by d.department\_name



6--les 5 employé les plus enciens

select \*

from(

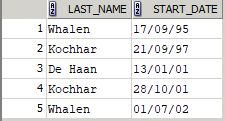
select e.last\_name, jh.start\_date

from employees e

join job\_history jh on jh.employee\_id=e.employee\_id

order by jh.start\_date )

where rownum <= 5



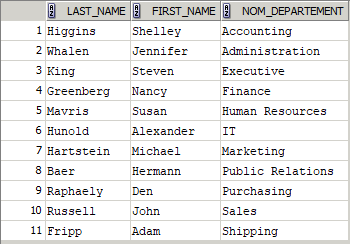
7--le nom et prenom de manager de chaque departement ---

Select e.LAST\_NAME, e.FIRST\_NAME, coalesce(d.DEPARTMENT\_NAME,'inconnu') nom\_departement

from employees e

join departments d on d.manager\_id=e.employee\_id

order by nom\_departement



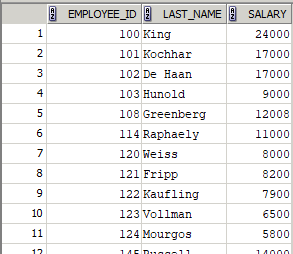
8--nom prenom de chaque manager et leur salaire --------

select distinct e1.employee\_id,e1.last\_name, e1.salary

from employees E1

join employees E2 on e1.employee\_id=e2.manager\_id

order by e1.employee\_id



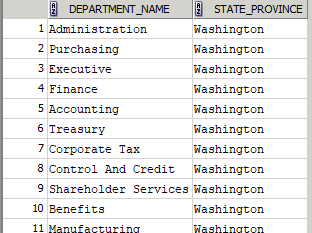
9--les departement qui sont a 'washington'----------------

select d.department\_name, l.state\_province

from departments d

join locations l on l.location\_id= d.location\_id

where l.state\_province = 'Washington'



10-- les cité et code postal de chaque location ou la cité contient 'South'

select postal\_code,city

from locations

where city like '%South%'



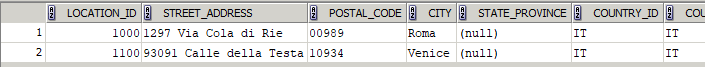
11--les locations en italie---

select \*

from locations l

join countries co on l.country\_id= co.country\_id

where co.country\_name = 'Italy'



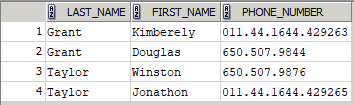
12--les numero de tel de employer qui ont dans leur nom taylor ou grant-----

select e.last\_name,e.first\_name ,e.phone\_number

from employees e

where e.last\_name like '%Taylor%' or e.last\_name like '%Grant%'

order by e.last\_name

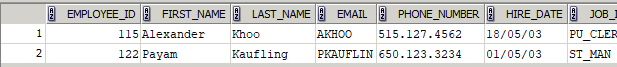


13--les employer embauché en mai 2003—

select distinct \* --e.last\_name,e.first\_name

from employees e

where extract( year from e.hire\_date) = 2003 and extract (month from e.hire\_date) = 05



14--les programmer manger par Alexander Hunold--------------------

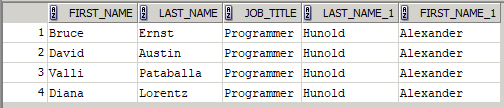
select e1.first\_name,e1.last\_name,j.job\_title,e2.last\_name,e2.first\_name

from employees e1

join employees e2 on e1.manager\_id= e2.employee\_id

join jobs j on j.job\_id =e1.job\_id

where j.job\_title = 'Programmer' and e2.first\_name = 'Alexander'



14--le min et max des salaires des sales manager--------------

select \*

from jobs

where job\_title= 'Sales Manager'



15--tous les employees d'europe qui gagne plus de 4500e--------

select e.first\_name,e.last\_name,e.salary, co.country\_name,r.region\_name

from employees e

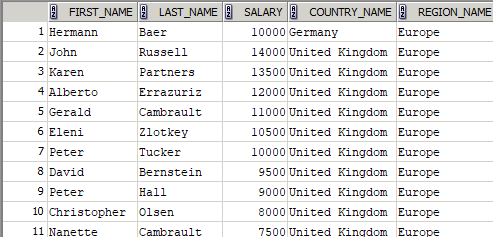
join departments d on d.department\_id=e.department\_id

join locations l on d.location\_id=l.location\_id

join countries co on l.country\_id=co.country\_id

join regions r on r.region\_id=co.region\_id

where r.region\_name= 'Europe' and e.salary > 4500



16--le nom et prénom, salaire et le pays de tous les employées d’Afrique et du moyen orient\*/

select e.first\_name,e.last\_name,e.salary, co.country\_name,r.region\_name

from employees e

join departments d on d.department\_id=e.department\_id

join locations l on d.location\_id=l.location\_id

join countries co on l.country\_id=co.country\_id

join regions r on r.region\_id=co.region\_id

where r.region\_name = 'Middle East and Africa'



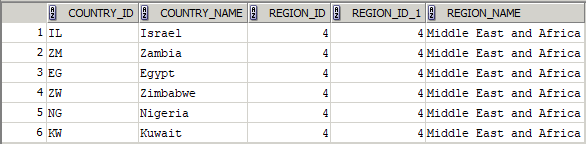
17--les pays de l’Afrique et du moyen orient \*/

select \*

from countries co

join regions r on co.region\_id= r.region\_id

where r.region\_name = 'Middle East and Africa'



18--le nombre d'employées en 'UK'\*/

select COUNT(e.employee\_id)

from employees e

join departments d on d.department\_id=e.department\_id

join locations l on d.location\_id=l.location\_id

join countries co on l.country\_id=co.country\_id

where country\_name = 'United Kingdom'



19--moyenne d'employées par continent\*/

select r.region\_name, avg(e.employee\_id)

from employees e

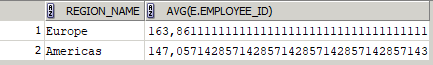
join departments d on d.department\_id=e.department\_id

join locations l on d.location\_id=l.location\_id

join countries co on l.country\_id=co.country\_id

join regions r on r.region\_id=co.region\_id

group by r.region\_name



20--le pourcentage des employées UK par rapport au employées de l’Europe\*/

select reu.region\_id,(reu.nbUK /reu.nbeurope)\*100

from (select cou.region\_id region\_id, count(\*) nbUK , Europe.nbeurope nbeurope

from employees e

join departments d on d.department\_id=e.department\_id

join locations l on d.location\_id=l.location\_id

join countries cou on l.country\_id=cou.country\_id

join (select r.region\_id region\_id , count (\*) nbeurope

from employees e

join departments d on d.department\_id=e.department\_id

join locations l on d.location\_id=l.location\_id

join countries co on l.country\_id=co.country\_id

join regions r on r.region\_id=co.region\_id

where r.region\_name = 'Europe'

group by r.region\_id ) Europe on Europe.region\_id = cou.region\_id

where cou.country\_name = 'United Kingdom'

group by cou.region\_id, Europe.nbeurope) reu

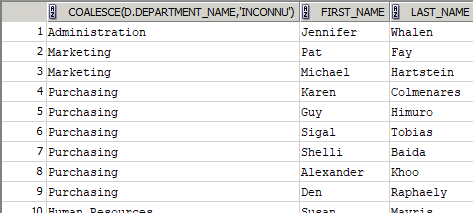


21--les départements de tous les employées\*/

select coalesce(d.department\_name,'inconnu'),e.first\_name,e.last\_name

from departments d

right join employees e on e.department\_id=d.department\_id;



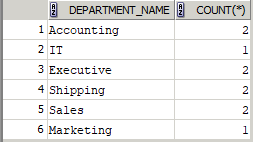
22-le nombre d'expérience de chaque departement \*/

select d.department\_name , count(\*)

from job\_history jh

join departments d on d.department\_id=jh.department\_id

group by d.department\_name;



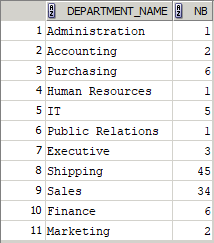
23--le nombre d'employées par département \*/

select d.department\_name, count(\*) nb

from employees e

join departments d on d.department\_id=e.department\_id

group by d.department\_name



24--la moyenne des employées par departement\*/

select avg(nb)

from(select d.department\_name, count(\*) nb

from employees e

join departments d on d.department\_id=e.department\_id

group by d.department\_name)



25--quels sont les départements qui ont un nombre d'employées inferieur a la moyenne

de nombre d'employées de tous les départements \*/

select d.department\_name, count(\*) nb

from employees e

join departments d on d.department\_id=e.department\_id

group by d.department\_name

having count(\*) <(select avg(nb)

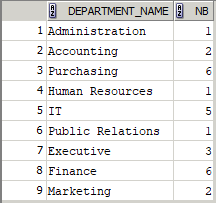
from(select d.department\_name, count(\*) nb

from employees e

join departments d on d.department\_id=e.department\_id

group by d.department\_name)

emd) ;



26--les manager qui vient de france ou UK\*/

select distinct e1.employee\_id,e1.last\_name

from employees E1

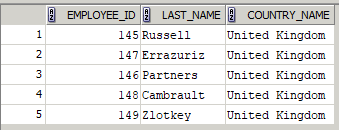
join employees E2 on e1.employee\_id=e2.manager\_id

join departments d on d.department\_id=e1.department\_id

join locations l on d.location\_id = l.location\_id

join countries cou on cou.country\_id= l.country\_id

where cou.country\_name = 'France' or cou.country\_name = 'United Kingdom'



27--quel sont les departement localiser au canada et en europe

SELECT dep.department\_name

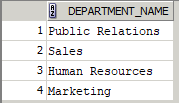
FROM DEPARTMENTS dep

join locations loc on dep.location\_id=loc.location\_id

join countries con on con.country\_id = loc.country\_id

join regions reg on con.region\_id=reg.region\_id

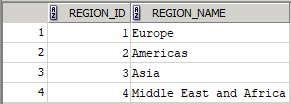
where reg.region\_name = 'Europe' or con.country\_name= 'Canada'



28--quel sont les region possible

select \*

from regions



29--le nombre de departement de chaque region

select reg.region\_name, count(dep.department\_id)

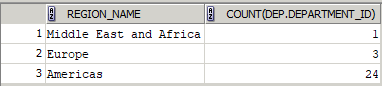
from departments dep

join locations loc on dep.location\_id=loc.location\_id

join countries con on con.country\_id = loc.country\_id

join regions reg on con.region\_id=reg.region\_id

group by reg.region\_name



30--verification :

select count(\*)

from departments



31--quel est la fonction de l'employer qui na pas de manager

select emp.first\_name, emp.last\_name,jo.job\_title

from employees emp

join jobs jo on emp.job\_id=jo.job\_id

where emp.manager\_id is null

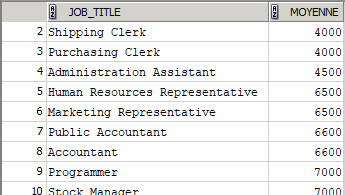


32--quel est la fonction la moins payer

select jo.job\_title,(jo.min\_salary+ jo.max\_salary )/2 as moyenne

from jobs jo

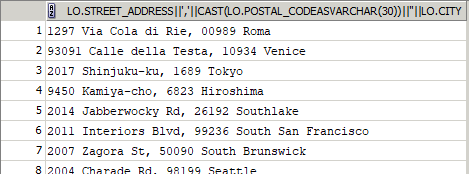
order by moyenne ;



33--afficher l'adresse des employées dans une seule colonne

select lo.street\_address || ', ' || cast(lo.postal\_code as varchar(30)) || ' ' || lo.city

from locations lo



34--la durée de travaille de chaque employée;

select em.last\_name, sysdate - em.hire\_date as datediff

from employees em

35--nombbre d'employées par pays

select c.country\_name,count(\*)

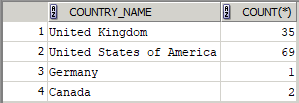
from Employees E

join departments D on e.department\_id = d.department\_id

join LOCATIONS L on D.location\_Id = L.location\_Id

join countries C on L.Country\_id = C.Country\_id

group by c.country\_name



36--quel est le nombre de departement dans chaque pays

select dep.department\_name , count(\*)

from employees emp

inner join departments dep on emp.department\_id=dep.department\_id

group by dep.department\_name

37--La durée de travail en année et mois

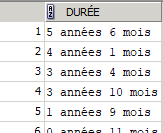
select (cast ((floor(months\_between (end\_date, start\_date) / 12))as varchar (2))

|| ' années ' ||

cast (floor(months\_between (end\_date, start\_date)) - (floor(months\_between (end\_date, start\_date) / 12) \*12) as varchar(2))

|| ' mois') as Durée

from job\_history;



38--Quels sont les employés des usa qui ont commencé à travailler après l'année 1990 ?

select em.first\_name, em.last\_name, jh.start\_date,con.country\_name

from employees em

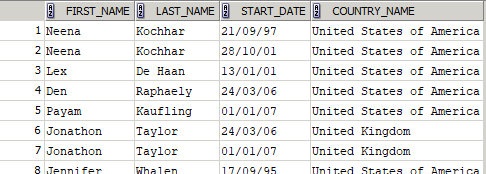
Join job\_history jh on em.employee\_id = jh.employee\_id

join departments de on jh.department\_id= de.department\_id

join locations lo on de.location\_id=lo.location\_id

join countries con on con.country\_id=lo.country\_id

Where extract (year from (start\_date)) > 1990



40-- Séparer les employés par tranche de salaire

select employee\_id, salary,

case

when salary <3000 then 'Faible'

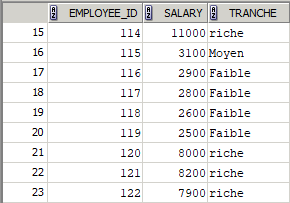
when salary between 3000 and 7000 then 'Moyen'

when salary between 7000 and 15000 then 'riche'

else 'super riche'

end as Tranche

from employees;



42--

select d.department\_name, e.employee\_id, e.first\_name, e.last\_name

from employees E

join Departments D on e.department\_id = d.department\_id

where d.department\_name = 'Administration' ;

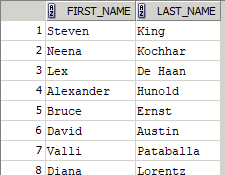


41--le nom les employés qui n'ont une commission ?

select first\_name,last\_name

from employees

where commission\_pct is null;



43--Quels sont les employés des usa qui ont commencé à travailler après l'année 1998 ?

select em.employee\_id, em.first\_name, em.last\_name, jh.start\_date,con.country\_name

from employees em

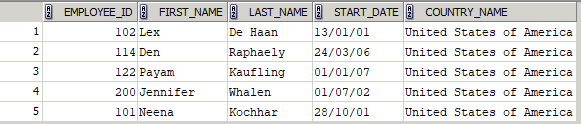
Join job\_history jh on em.employee\_id = jh.employee\_id

join departments de on jh.department\_id= de.department\_id

join locations lo on de.location\_id=lo.location\_id

join countries con on con.country\_id=lo.country\_id

Where extract (year from (start\_date)) > 1998 and con.country\_name= 'United States of America';



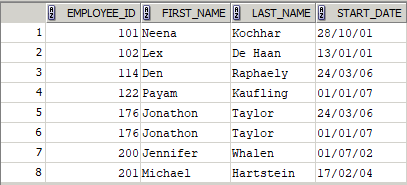
44--Quels sont les employés de l'historique qui ont commencé à travailler après l'année 1998 ?

select e.employee\_id, e.first\_name, e.last\_name, start\_date

from EMPLOYEES E

Join job\_history JH on e.employee\_id = jh.employee\_id

Where extract (year from (start\_date)) > 1998;



45--le nombre de programmeur

select count(\*)

from employees em

join jobs jo on jo.job\_id=em.job\_id

where jo.job\_title = 'Programmer'



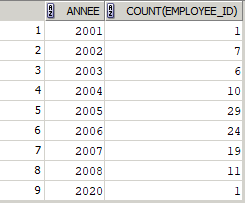
46-- le nombre d'employés embauchés dans chaque année ?

select extract(year from (hire\_date)) as annee, count(employee\_id)

from employees

group by extract(year from (hire\_date))

order by annee;



47--le nombre de Sales Representative par departement

select dep.department\_name, count(em.employee\_id)

from departments dep

join employees em on dep.department\_id=em.department\_id

join jobs jo on jo.job\_id=em.job\_id

where jo.job\_title = 'Sales Representative'

group by dep.department\_name



48--le nombre d'employées dans le departement shipping

select count(\*)

from employees em

join departments dep on dep.department\_id=em.department\_id

where dep.department\_name = 'Shipping'



50--la moyenne salaire d'employées dans le departement shipping

select avg (em.salary)

from employees em

join departments dep on dep.department\_id=em.department\_id

where dep.department\_name = 'Shipping';



51--les employées du departement shipping qui qu'on un slaire inferieur a la moyenne des salaires shipping

select em.first\_name,em.last\_name,em.salary ,dep.department\_name

from employees em

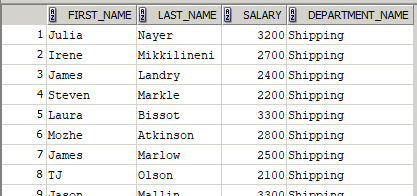
join departments dep on dep.department\_id=em.department\_id

where dep.department\_name = 'Shipping' and em.salary < (select avg (em.salary)

from employees em

join departments dep on dep.department\_id=em.department\_id

where dep.department\_name = 'Shipping')



53--le salaire moyen de chaque ville

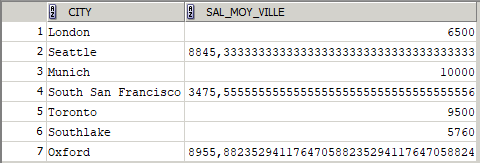
select loc.city , avg (salary) as sal\_Moy\_Ville

from employees Em

join Departments dep on em.department\_id = dep.department\_id

join locations Loc on dep.location\_id = loc.location\_id

group by loc.city;



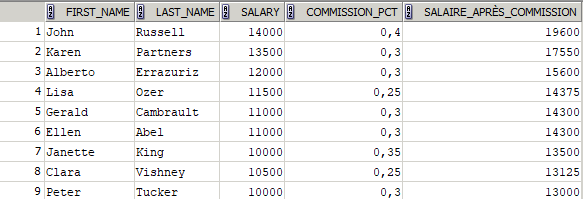
54--le salaire des employées qui ont une commision

select em.first\_name, em.last\_name, em.salary, em.commission\_pct, (em.salary + (em.salary \* commission\_pct)) as salaire\_après\_commission

from employees em

where commission\_pct is not null

order by salaire\_après\_commission desc;

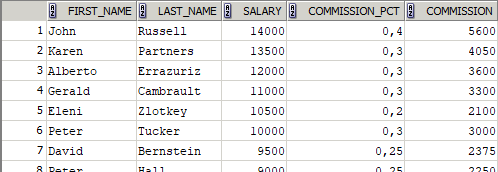


56--le motant de la commision pour chaque employées

select first\_name, last\_name, salary, commission\_pct,(salary \* commission\_pct) as Commission

from employees

where commission\_pct is not null;

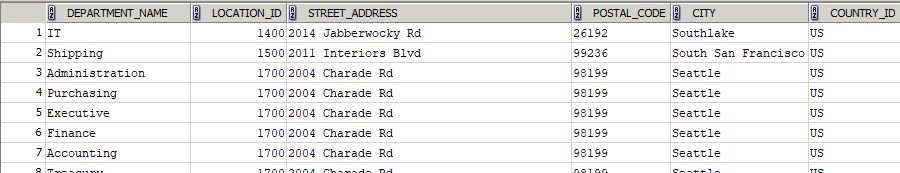


57--tous les departements et leurs locations

select dep.department\_name,lo.location\_id ,lo.street\_address,lo.postal\_code,lo.city,lo.country\_id

from departments dep

join locations lo on lo.location\_id= dep.location\_id



58--le departement qui a plus d'employées

select \*

from (select dep.department\_name, count(\*) nbr

from employees em

join departments dep on dep.department\_id=em.department\_id

group by dep.department\_name

order by nbr desc)

where rownum <= 1



59--les employee et leur debut et fin de contrat et leur departement

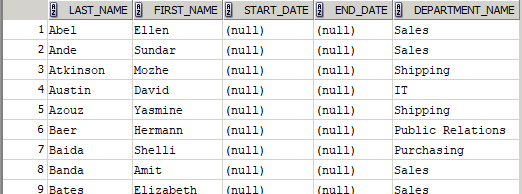
select distinct em.last\_name,em.first\_name , jh.start\_date,jh.end\_date ,dep.department\_name

from departments dep

join employees em on dep.department\_id=em.department\_id

left join job\_history jh on em.employee\_id=jh.employee\_id

order by em.last\_name



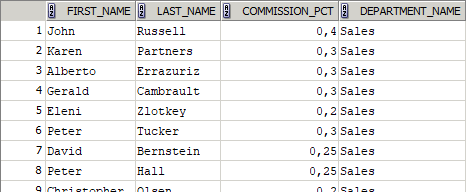
60--les employees du deartement sales qui ont une commisson

select em.first\_name ,em.last\_name,em.commission\_pct,dep.department\_name

from employees em

join departments dep on dep.department\_id=em.department\_id

where em.commission\_pct is not null and dep.department\_name ='Sales';



61--les employée des usa qui ont commencé apres 2006

select e.first\_name, e.last\_name, e.hire\_date

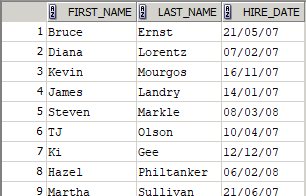
from employees e

join departments d on d.department\_id=e.department\_id

join locations l on d.location\_id=l.location\_id

join countries co on l.country\_id=co.country\_id

Where co.country\_name = 'United States of America' and extract (year from (e.hire\_date)) > 2006 ;

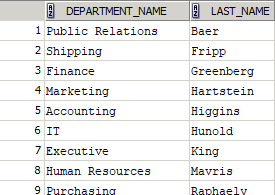


63--tous departement et leur mananger

select dep.department\_name, em.last\_name

FROM DEPARTMENTS dep

left join employees em on dep.manager\_id=em.employee\_id



64--Quelle est la moyenne de salaires pour chaque fonction ?

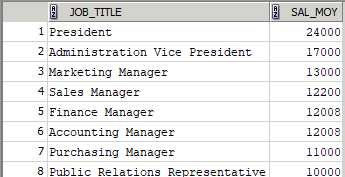
select job\_title, avg (salary) as sal\_moy

from employees em

Join jobs J on em.job\_id = j.job\_id

group by j.job\_title

order by sal\_moy desc;



65--tous les managers et leur departement

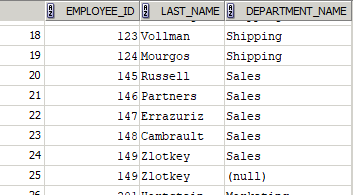
select distinct e1.employee\_id,e1.last\_name,dep.department\_name

from employees E1

join employees E2 on e1.employee\_id=e2.manager\_id

left join departments dep on dep.department\_id=e2.department\_id

order by e1.employee\_id



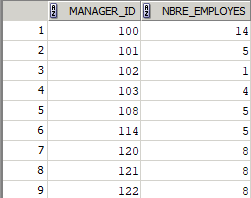
66--Combiens d'employés rattachés au meme manager ?

select manager\_id, count(employee\_id) as Nbre\_employes

from employees

where manager\_id is not null

group by manager\_id;



67-- le premier employées embauchée

select \*

from(select employee\_id, first\_name, last\_name,extract(year from (hire\_date)) as Année

from employees

order by Année)

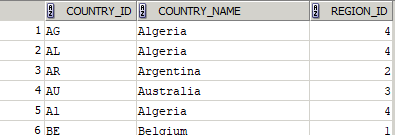
where rownum <= 1;



68--ajouter un pays.

INSERT INTO COUNTRIES (COUNTRY\_ID, COUNTRY\_NAME, REGION\_ID)

VALUES ( 'AL', 'Algeria', 4);



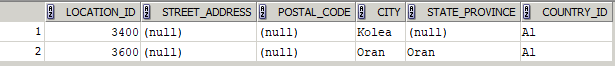
69--Ajouter une location

INSERT INTO LOCATIONS (LOCATION\_ID,STREET\_ADDRESS,POSTAL\_CODE,CITY,STATE\_PROVINCE,COUNTRY\_ID)

VALUES(3400,null,null,'Kolea',null,'Al');

INSERT INTO LOCATIONS (LOCATION\_ID,STREET\_ADDRESS,POSTAL\_CODE,CITY,STATE\_PROVINCE,COUNTRY\_ID)

VALUES(3600,null,null,'Oran','Oran','Al');



70--Ajouter un departement

INSERT INTO DEPARTMENTS( DEPARTMENT\_ID, DEPARTMENT\_NAME, MANAGER\_ID, LOCATION\_ID )

VALUES(280,'DevOps',null,3400);



**71**--Ajouter un employé

INSERT INTO EMPLOYEES(EMPLOYEE\_ID,FIRST\_NAME,LAST\_NAME,EMAIL,PHONE\_NUMBER,HIRE\_DATE,JOB\_ID,SALARY,COMMISSION\_PCT,MANAGER\_ID,DEPARTMENT\_ID)

VALUES(207,'Yasmine','Azouz','founas','0766805704',TO\_DATE('20200928','yyyymmdd'),'IT\_PROG',null,null,null,50);



72--le nombre des employés dans le départemet de RH

select first\_name, last\_name, d.department\_name

from employees e

join departments d on e.department\_id = d.department\_id

where lower(department\_name) like 'human resources';



73-Quel est le salaire maximal ?

select max(salary) as Sal\_Max

from employees;

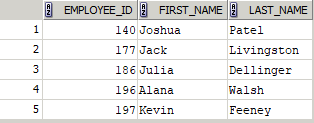


74--Quels sont les employés embauché le deuxieme trimestre de 2006 ?

select employee\_id, first\_name,last\_name

from employees

where extract (month from (hire\_date)) in (4,5,6) and extract (year from (hire\_date)) = 2006;

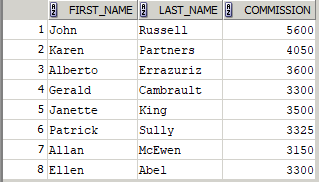


75--Quels sont les employés qui ont une commission > 3000

select first\_name, last\_name,(commission\_pct\*salary) as commission

from employees

where (commission\_pct\*salary)> 3000;



76-Quels sont les villes de départements qui se trouvent aux états unis

select distinct city, country\_name

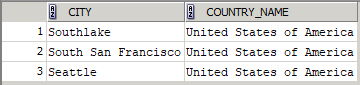
from employees E

join Departments D on e.department\_id = D.department\_id

join locations L on d.location\_id = l.location\_id

join countries C on l.country\_id = C.country\_id

where lower(country\_name) = 'united states of america';



77--Les email des employés des Etats unis

select Email, country\_name

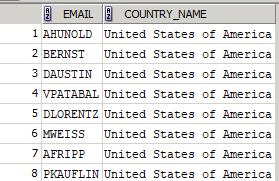
from employees E

join Departments D on e.department\_id = D.department\_id

join locations L on d.location\_id = l.location\_id

join countries C on l.country\_id = C.country\_id

where lower(country\_name) = 'united states of america';



78--Tous les employés de département administration classés par ancienneté

select first\_name, last\_name, d.department\_name, hire\_date

from employees em

join departments d on em.department\_id = d.department\_id

where lower(department\_name) like '%administration'

order by hire\_date;



79--Quels sont les employés qui ont la commission la plus importante ?

select first\_name,last\_name, commission\_pct

from employees

where commission\_pct is not null and commission\_pct = (select max(commission\_pct) from employees);



80--Quelles sont les fonctions qui correspondent à la commission la plus importante ?

select job\_title, commission\_pct

from employees E

join Jobs J on e.job\_id = j.job\_id

where commission\_pct is not null and commission\_pct = (select max(commission\_pct) from employees);



81--les employés qui sont embauchés le meme jour

select distinct e.employee\_id, e.first\_name, e.last\_name, e.hire\_date

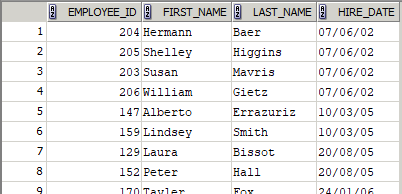
from employees E

join employees E1 on e.hire\_date=e1.hire\_date

group by e.employee\_id, e.first\_name, e.last\_name,e.hire\_date

having count(e.hire\_date) > 1

order by e.hire\_date;

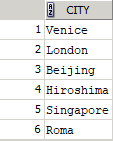


82--Quelles sont les villes dont l'état de province n'est pas renseigné

select distinct city

from locations

where state\_province is null;



83--Les département de texas

select d.department\_name, state\_province

from departments d

join locations l on d.location\_id = l.location\_id

where lower(state\_province)= 'texas';



84--Quel est le salaire du président

select j.job\_title, salary

from employees E

join jobs J on E.job\_id = j.job\_id

where lower(j.job\_title) = 'president';



85--Combien y' a-t-il de comptables?

select count (j.job\_title) as Nombre\_de\_comptable

from employees E

Join jobs J on e.job\_id = j.job\_id

where j.job\_title = 'Accountant';



86--Combien y' a-t-il de comptables par département?

select d.department\_name, count(j.job\_title) as Nombre\_de\_comptable

from employees E

Join jobs J on e.job\_id = j.job\_id

join departments D on e.department\_id = d.department\_id

where j.job\_title = 'Accountant'

group by d.department\_name;



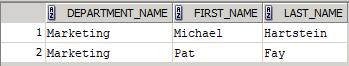
87--Quels sont les employés de département marketing ?

select d.department\_name,e.first\_name, e.last\_name

from employees E

join Departments D on e.department\_id = d.department\_id

where d.department\_name = 'Marketing' ;

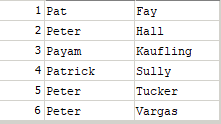


89--Quels sont les employés dont le nom(prénom) commence par 'P'?

select first\_name,last\_name

from employees

where first\_name like 'P%';

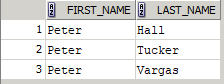


90--les employés qui s'appellent 'peter'

Select first\_name, last\_name

from employees

where lower(first\_name) = 'peter';



91--Quel est le nombre de département dans chaque région ?

select r.region\_name, count(d.department\_id)

from departments D

join LOCATIONS L on D.location\_Id = L.location\_Id

join countries C on L.Country\_id = C.Country\_id

join Regions R on c.region\_id = r.region\_id

group by r.region\_name;



92--Quelles sont les villes où on peut trouver des départements aux Etats unis ?

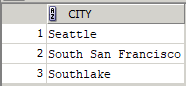
select distinct l.city

from departments D

join LOCATIONS L on D.location\_Id = L.location\_Id

join countries C on L.Country\_id = C.Country\_id

Where c.country\_name = 'United States of America';



93--les employés qui ont le même nom de famille

select distinct e.first\_name, e.last\_name

from employees E

cross join employees E1 where e.last\_name= e1.last\_name

group by e.first\_name, e.last\_name

having count(e.last\_name)>1

order by e.last\_name;

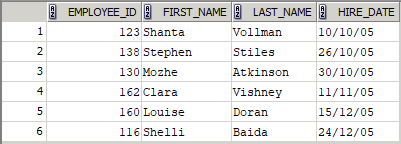
94--Les employés embauchés le dernier semestre de 2005

select employee\_id, first\_name, last\_name, hire\_date

from employees

where (extract (month from(hire\_date)) between 10 and 12 and extract (year from(hire\_date))= 2005)

order by hire\_date;



95--Quel est le nombre de villes où se trouve des départements dans chaque pays ?

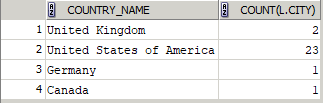
select c.country\_name, count(l.city)

from departments D

join LOCATIONS L on D.location\_Id = L.location\_Id

left join countries C on L.Country\_id = C.Country\_id

group by c.country\_name;



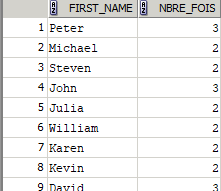
96Quel est le nombre des employés qui ont le même nom?

select first\_name, count(first\_name) as Nbre\_fois

from employees

group by first\_name

having count(first\_name)>1 ;

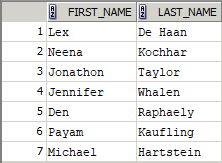


97--Les employés qui ont quitté leur travail

select distinct first\_name, last\_name

from job\_history jh

join employees e on jh.employee\_id = e.employee\_id;



98--Quels sont les pays où il n'y a pas de départements ?

select co.country\_name

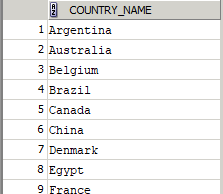
from countries Co

left join locations lo on Co.country\_id = lo.country\_id

left join Departments D on lo.location\_id = D.location\_id

where department\_name is null

order by Co.country\_name;



99--Quelle est la fonction ayant un nombre plus important des employés

select job\_title, Nbre\_emp

from (select jo.job\_title, count(\*) as Nbre\_emp

from employees Em

join jobs Jo on Em.job\_id = jo.job\_id

group by jo.job\_title

order by Nbre\_emp DESC)

where rownum =1 ;

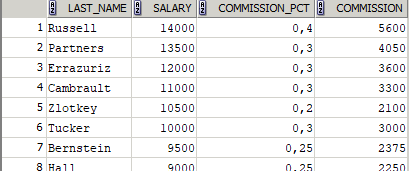


100--Calculer la valeur de la commission

select last\_name, salary, commission\_pct,(salary \* commission\_pct) as Commission

from employees

where commission\_pct is not null;



101-localisation dont le code postale est absent

select \*

from locations

where postal\_code is null;

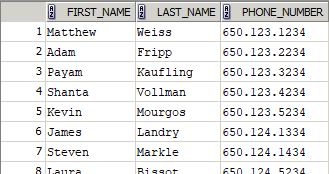


102-Les employés ayant un numéro de téléphone qui se termine par 34

select first\_name, last\_name, phone\_number

from employees

where substr(phone\_number,length(phone\_number)-1, 2)=34;



# TP SQL Server - GYMNASE

1--------------------LISTE DE LA TABLE SEANCES----------

SELECT\*

FROM Seances;

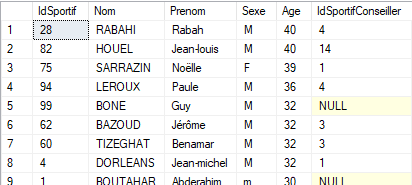


2-------------LISTE DES SPORTIFS PAR ORDRE DECROISSANT D'AGE--------------

SELECT \*

FROM Sportifs

ORDER BY AGE DESC; -- décroissant

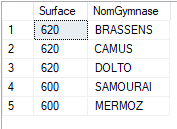


3------------Lister les 5 gymnases les plus grands----------------------\*/

SELECT TOP 5 Surface, NomGymnase

FROM Gymnases

ORDER BY Surface DESC;



5-----------LISTE DES SPORTIF (NOM PRENOM) AGEE DE PLUS DE 30 ANS

SELECT Nom, Prenom, Age

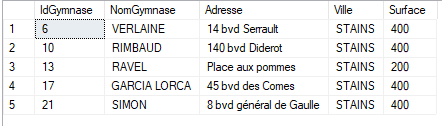
FROM Sportifs WHERE Age> 30;



5-- Lister les gymnases de la ville de "Stains"--

SELECT\*

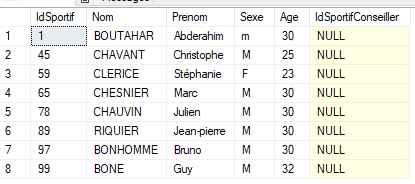
FROM Gymnases WHERE Ville='Stains';



6----Lister les sportifs n'ayant pas de conseiller--------\*/

SELECT\*

FROM Sportifs where IdSportifConseiller is NULL ;



7/\*Lister les sports pratiqués (uniquement le libellé de chaque sport)\*/

SELECT Libelle

from Sports;



8/\*Lister les différentes valeurs de Sexe possible\*/

SELECT DISTINCT SEXE

FROM Sportifs;



9/\*Afficher en heure la durée de chaque séance (stockée en minute dans la table)\*/

SELECT DISTINCT Duree,

round(Duree/60,2) as 'duree heure'

FROM Seances

ORDER BY 1;

10-select SEC\_TO\_TIME(30) --Durée\*60) from Seances – marche qu’on a des secondes.

11/\* donner pour chaque sportif un prefixe de courtoisie de Mm pour les filles et M. pour les garcons

select sexe, case

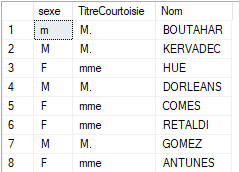
when Sexe= 'm' then 'M.'

else 'mme'

end as TitreCourtoisie

, Nom

from Sportifs ;



12/\*créer une nouvelle colonne qui prend la valeur petite, moyen, grand selon la surface d'un gymnase\*/

select NomGymnase, case

when Surface> 550 then 'grand'

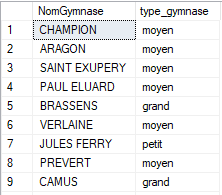
when Surface<400 then 'petit'

when Surface between 400 and 500 then 'moyen'

else 'inconnu'

end as type\_gymnase

from Gymnases ;



13/\*écrire en minuscule la concaténation du nom et 1ere lettre du prénom\*/

select lower(Nom + '.' + LEFT (prenom,1)) as nomP

from Sportifs

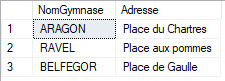


14/\*les gymnases qui sont situé sur des « place »

select NomGymnase, Adresse

from Gymnases

where Adresse like '%place%' ;



15 /\*donner la date d'aujourd'hui\*/

select GETDATE() ;



16/\* compter le nombre de sportifs\*/

select count(IdSportif) nombre\_sportifs

from Sportifs



17/\* les sportifs qui ont un conseiller\*/

select \*

from Sportifs

where IdSportifConseiller is not null



18/\*le nombre de ville distincte \*/

select count (distinct Ville) nombre\_ville

from Gymnases



19/\*la surface moyenne des gymnases \*/

select avg(surface) as moyenne\_surface

from Gymnases ;



20/\*age moyen age min et age max des sportif \*/

select AVG(age) as agemoyen,min(age) as doyen,max(age) as bingamin

from Sportifs



21/\*nombre de sportif par sexe\*/

select sexe, count (sexe) nbr\_par\_sexe

from Sportifs

group by Sexe



21/\*pour chaque ville le plus petit et grand gymnase \*/

select ville,min(surface) as le\_petit, max(surface) as le\_grand

from gymnases

group by ville ;



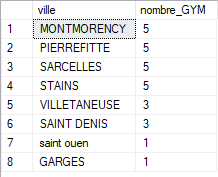
23/\* la ville qui a plus de gymnase \*/

select ville,count(Ville) as nombre\_GYM

from Gymnases

group by Ville

order by nombre\_GYM desc ;

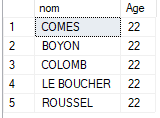


24/\*les 5 sportifs les plus âgés \*/

select top 5 nom, Age

from Sportifs

order by Age



25/\* les gymnases qui ont une surface plus de 500m\*/

select distinct Ville

from Gymnases

where Surface > 500 ;

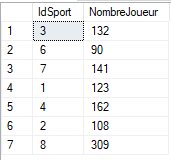


26/\*le nombre de joueur par sport \*/

select IdSport, count(IdSportif) as NombreJoueur

from Jouer

group by IdSport ;



27--junior senior1 senior2---------------

select count(\*) , case

when age between 20 and 24 then 'junior'

when age between 25 and 30 then 'senior 1'

when age between 31 and 45 then 'senior 2'

else 'inconnu'

end AS cat

from Sportifs

group by case

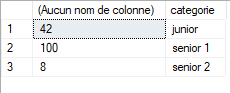
when age between 20 and 24 then 'junior'

when age between 25 and 30 then 'senior 1'

when age between 31 and 45 then 'senior 2'

else 'inconnu'

end

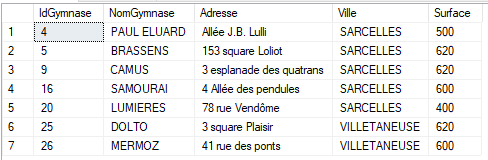


28/\* tous le gymnase de Villetaneuse ou de sarcelles qui ont une surface de plus de 400m\*/

select \*

from Gymnases

where Surface > 400 and Ville ='villetaneuse' or Ville = 'sarcelles'



29/\*nom et prenom de tous les joueur de handball\*/

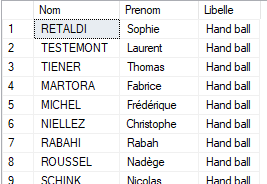
select Nom,Prenom, Libelle

from Sportifs S

join Jouer J on s.IdSportif=j.IdSportif

join Sports S1 on S1.IdSport=j.IdSport

where s1.Libelle = 'Hand ball'



30/\*les seances dans handball dans quel gymnase et leurs horaires \*/

select Sp.Libelle,Sc.Jour, g.NomGymnase, Sc.Horaire

from Gymnases G

join Seances Sc on sc.IdGymnase=G.IdGymnase

join Sports Sp on Sc.IdSport=Sp.IdSport

where Sp.Libelle = 'Hand ball'



30/\*les gymnase qui ont des seance de hockey mercredi après 15h \*/

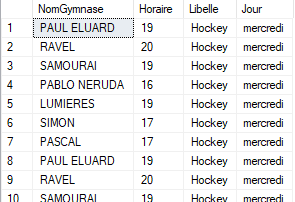
select g.NomGymnase, Sc.Horaire,sp.Libelle,sc.Jour

from Gymnases G

join Seances Sc on sc.IdGymnase=G.IdGymnase

join Sports Sp on Sc.IdSport=Sp.IdSport

where sp.Libelle ='hockey' and sc.Horaire > 15 and sc.Jour ='mercredi' ;



31/\* les gymnase qui propose des seance de basket ball ou de volley ball\*/

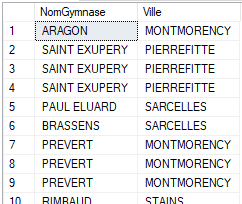
select g.NomGymnase ,g.Ville

from Gymnases G

join Seances Sc on sc.IdGymnase=G.IdGymnase

join Sports Sp on Sc.IdSport=Sp.IdSport

where sp.Libelle ='basket ball' or sp.Libelle= 'volley ball' ;

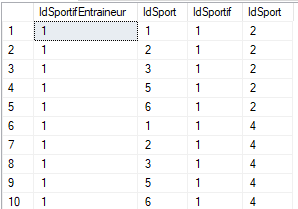


32/\*les entraineurs qui sont aussi joueurs \*/

select \*

from Entrainer e

left join Jouer j on j.IdSportif=e.IdSportifEntraineur ;



33/\* les sportifs qui sont conseillers\*/

select \*

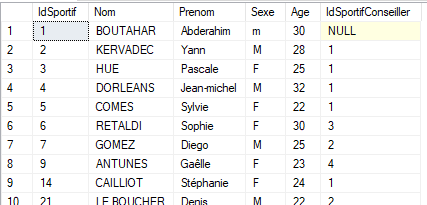
from Sportifs s1

34

select distinct \*

from Sportifs s1

where s1.IdSportif in (select s2.IdSportifConseiller from Sportifs s2) ;



35/\*le nom du conseiller du sportif 'kervadec'\*/

select IdSportif, Nom,Prenom

from Sportifs Conseiller

where Conseiller.IdSportif in (select s2.IdSportifConseiller from Sportifs s2 where Nom ='kervadec')



36/\*les entraineur qui entraine le basket et le hand ball \*/

Jointure

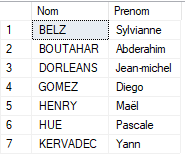
select distinct Nom,Prenom

from Sportifs Sf

join Entrainer E on e.IdSportifEntraineur=SF.IdSportif

join Sports Sp on Sp.IdSport=E.IdSport

where Sp.Libelle in ('Hand ball', 'Basket ball')



37--2eme méthodes

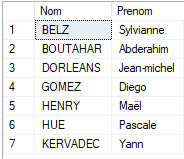
select distinct Nom, Prenom

from Sportifs, Entrainer, Sports

where Sportifs.IdSportif = Entrainer.IdSportifEntraineur

and Entrainer.IdSport = Sports.IdSport

and Libelle in ('Hand ball', 'Basket ball');



39/\*moyenne d’age des joueur de basket ball et le min et le max \*/

select AVG(age) as moyenne\_age,min(age) min\_age,max(age) max\_age

from Sportifs SF

join Jouer J on j.IdSportif=sf.IdSportif

join Sports Sp on sp.IdSport=j.IdSport

where sp.Libelle ='basket ball'

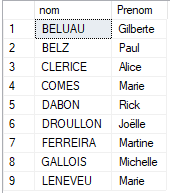


40/\* les sportif qui ne pratiquent aucun sport \*/

select distinct nom,Prenom

from Sportifs SF

where IdSportif not IN (select IdSportif from Jouer)



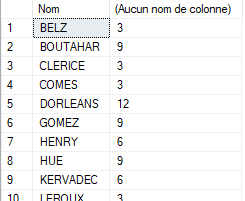
41/\*le nombre sport arbitré par chaque sportifs \*/

select sf.Nom,count(ar.IdSport)

from Sportifs Sf

join Arbitrer Ar on Sf.IdSportif= Ar.IdSportif

group by(sf.Nom)



42/\*les gymnase qui plus de 9 seance le mercredi \*/

select NomGymnase, count(Horaire)

from Seances S

join Gymnases G on g.IdGymnase=s.IdGymnase

where Jour = 'mercredi'

group by NomGymnase

having count(horaire) > 9



43/\*dans quel gymnase et quel jour y’a-t-il plus de 4 séances de volley dans la journée\*/

select nomgymnase,sp.Libelle,count(horaire)

from Gymnases g

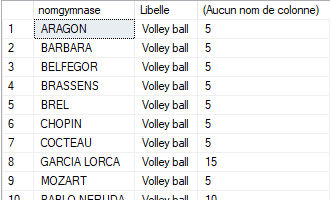
join Seances s on g.IdGymnase=s.IdGymnase

join Sports Sp on s.IdSport= Sp.IdSport

where Libelle ='volley ball'

group by NomGymnase,sp.Libelle

having count(Horaire) >= 4



44/\* chaque entraineur de hand ball quel nombre de séances journalière qu’il assure\*/

select Sf.Nom, sp.Libelle, count(Horaire)

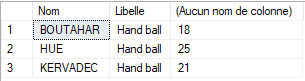
from Sportifs Sf

join Seances Sc on Sc.IdSportifEntraineur=Sf.IdSportif

join Sports Sp on sp.IdSport=sc.IdSport

where Libelle ='hand ball'

group by Sf.Nom, Libelle



45/\* nombre de seance journaliere de chaque sport des gymnases de montmorency \*/

select sp.Libelle, nomgymnase, count(sc.IdSport)

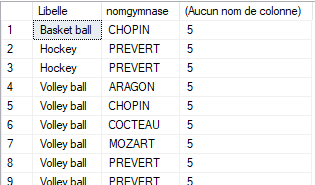
from Gymnases g

join Seances Sc on sc.IdGymnase = G.IdGymnase

join Sports Sp on Sp.IdSport=Sc.IdSport

where g.Ville ='montmorency'

group by sp.Libelle, NomGymnase,jour



46/\*Solution 2\*/

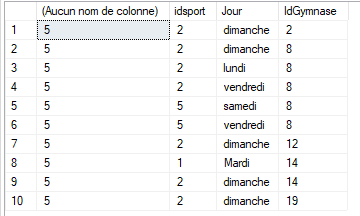
select count( sc.IdGymnase),idsport, Jour,sc.IdGymnase

from Seances sc

join Gymnases G on g.IdGymnase= sc.IdGymnase and sc.IdGymnase in (2,8,12,14,19)

group by idsport, Jour, sc.IdGymnase

order by sc.IdGymnase



47/\* le sport pratiqué par le joueur le plus jeune\*/

select Libelle, min(age) min\_age

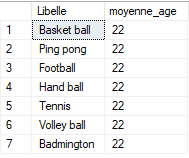
from Sports Sp

join Jouer j on j.IdSport= sp.IdSport

join Sportifs Sf on j.IdSportif=sf.IdSportif

group by sp.Libelle

order by min\_age



48/\* la moyenne d’âge de chaque sport\*/

select Libelle, avg(age) moyenne\_age

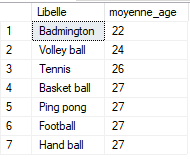
from Sports Sp

join Jouer j on j.IdSport= sp.IdSport

join Sportifs Sf on j.IdSportif=sf.IdSportif

group by sp.Libelle

order by moyenne\_age

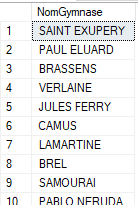


49/\*le gymnase qui n’ont pas de seance le dimanche\*/

select NomGymnase

from Gymnases G

where IdGymnase not in (select IdGymnase from Seances where Jour ='dimanche') ;



50/\*les horaire des 1er et derniere seances de chaque gymnase de Saints\*/

select NomGymnase,Jour, min(horaire),max(horaire)

from Seances Sc

join Gymnases G on g.IdGymnase= Sc.IdGymnase

where g.Ville = 'stains'

group by NomGymnase,Jour



51/\*les grand gymnase de chaque ville \*/

select distinct Ville,max(surface) grand,NomGymnase--,max(NomGymnase)

from Gymnases

group by ville,NomGymnase

order by Ville, grand desc

/\*Dans une même requête, donner pour chaque sportif (nom et prénom)

• le nombre de sports qu'il joue

• le nombre de sports qu'il arbitre

• le nombre de sports qu'il entraîne

• on doit avoir tous les sportifs, avec des 0 si besoin\*/------------------------------------------------

52-----

select distinct nom,prenom ,sp.Libelle--count(Nom)--, count(Ar.idsport),count (e.idsport)

from Sportifs Sf

join Jouer J on j.IdSportif=Sf.IdSportif

join Sports Sp on sp.idsport= j.IdSport

--group by Nom

order by Nom

join Arbitrer Ar on Ar.IdSportif=Sf.IdSportif

join Entrainer E on e.IdSportifEntraineur=Sf.IdSportif

group by ar.IdSport,e.IdSport,j.IdSport

# TP BIBLIOTHEQUE

1-- créer une base de données

create database bibliothèque

-----------------création de table------------------------------

2/\* creation de table Editeur\*/

CREATE TABLE Editeur

(

IdEditeur INT NOT NULL identity,

Nom VARCHAR (30),

Telephone VARCHAR(15),

Pays VARCHAR(30),

CONSTRAINT PK\_Editeur primary key (IdEditeur)

);

4/\*création de table MotCle\*/

CREATE TABLE MotCles

(

IdMot VARCHAR(50) NOT NULL ,

Mot VARCHAR (50),

Descriptif VARCHAR(30),

CONSTRAINT PK\_MotCles primary key (IdMot)

);

5/\* création de table Ouvrage\*/

CREATE TABLE Ouvrage

(

IdOuvrage INT NOT NULL identity ,

Titre VARCHAR (30),

IdEditeur INT NOT NULL,

Annee INT ,

CONSTRAINT PK\_Ouvrage primary key (IdOuvrage),

CONSTRAINT FK\_Editeur FOREIGN KEY (IdEditeur) REFERENCES Editeur(IdEditeur)

);

6/\* creation de table Organisme\*/

CREATE TABLE Organisme

(

IdOrganisme INT NOT NULL identity ,

Nom VARCHAR (50),

Telphone\_Oranisateur VARCHAR(50),

Pays\_Organisateur VARCHAR(50),

CONSTRAINT PK\_Organisme primary key (IdOrganisme), -- declarer une clé primaire

);

7/\* création de table Auteur\*/

CREATE TABLE Auteur

(

IdAuteur INT NOT NULL identity ,

NomAuteur VARCHAR (50),

Email VARCHAR(50),

SiteWeb VARCHAR(50),

CONSTRAINT PK\_Auteur primary key (IdAuteur),

);

8 /\* création de table Revue\*/

CREATE TABLE Revue

(

IdRevue INT NOT NULL identity ,

Nom\_Revue VARCHAR (50),

IdEditeur INT NOT NULL ,

Annee\_creation INT,

primary key (IdRevue),

foreign key (IdEditeur) REFERENCES Editeur(IdEditeur)

);

9/\*Creation de Table Article\*/

CREATE TABLE Article

(

IdArticle INT NOT NULL,

IdRevue INT NOT NULL,

Titre VARCHAR(20),

PageDebut INT,

PageFin INT,

Volume INT,

Serie VARCHAR(50),

primary key (IdArticle),

foreign key (IdRevue) REFERENCES Revue(IdRevue)

);

10/\* creation de table Article\_Auteur \*/

CREATE TABLE Article\_Auteur

(

IdArticle INT NOT NULL,

IdAuteur INT NOT NULL ,

foreign key (IdArticle) REFERENCES Article(IdArticle),-- declaration de clé etrangere

foreign key (IdAuteur) REFERENCES Auteur(IdAuteur)

);

/\* creation de table Article\_MotCles\*/

CREATE TABLE Article\_MotCles

(

IdArticle INT NOT NULL,

IdMot VARCHAR(50) NOT NULL,

foreign key (IdArticle) REFERENCES Article(IdArticle),

foreign key (IdMot) REFERENCES MotCles(Idmot)

);

11/\* creation de table Auteur\_Organisme\*/

CREATE TABLE Auteur\_Organisme

(

IdAuteur INT NOT NULL,

IdOrganisme INT NOT NULL,

foreign key (IdOrganisme) REFERENCES Organisme(IdOrganisme),

foreign key (IdAuteur) REFERENCES Auteur(IdAuteur)

);

12/\* Creation de Table Ouvrage\_Auteur \*/

CREATE TABLE Ouvrage\_Auteur

(

IdAuteur INT NOT NULL,

IdOuvrage INT NOT NULL,

foreign key (IdOuvrage) REFERENCES Ouvrage(IdOuvrage),

foreign key (IdAuteur) REFERENCES Auteur(IdAuteur)

);

13---insertion-----------------------------

insert into Auteur(NomAuteur,Email,SiteWeb)

values ('yasmine','yasmine-azouz@gmail.com','journalyasmine.com');

14--

select\*

from Auteur

15--

insert into Auteur(NomAuteur)

values ('mouloud mammeri');

16--

select\*

from Auteur

17--

delete from Auteur

where NomAuteur = 'mouloud mammeri'

18--

select\*

from Editeur

19-

insert into Editeur ( Nom,Telephone,Pays)

values ('meriame','0677551909','france'),

('nadia','0612155236','espagne'),

('Dyhia','0766805704','algerie'),

('lynda','0670131504','france'),

('samir','0612138004','italie'),

('samir',NULL,NULL);

select\*

from Editeur

20 /\* modifier le nom éditeur en les mettant la 1er lettre en MAJ et le reste en Min \*/

select initcap (Nom)

from Editeur

21-

SELECT idEditeur, initcap(nom)

FROM Editeur

22/\* la liste des nom des editeur\*/

select nom from Editeur;

23/\*les pays des editeur sans doublant\*/

select distinct pays

from Editeur;

23--

insert into Article( IdArticle, idRevue,Titre,PageDebut,PageFin,Volume,Serie)

values (1,2,'le monde', 2,6,1,1),

(2,2,'news locale', 1,1,2,1),

(3,4,'paparatzi', 4,6,3,2),

(4,3,'celebrite', 3,10,4,4),

(5,1,'film', 6,12,3,5),

(6,1,'sport', 5,18,5,3);

24/\* lister les titres des article par ordre alphabethique\*/

select Titre

from Article

order by Titre asc

/

select \* from Revue;

25--

insert into Revue(nom\_Revue,idEditeur,Annee\_Creation)

values('le soleil',1,2002),

( 'prada',1,2010),

( 'people',4,2008),

( 'Deep',3,2002),

( 'planet',2,2014),

( 'planet',4,2011);

26 /\* insérer des données dans la table Ouvrage \*/

insert into ouvrage

values ( 'in die digitale bildverare',1 ,1994),( 'data structure and algorithme',2,1983)

('tous sur c++',3,1992),('sql facil',2,2006);

27—

select\* from ouvrage -- verification que la table a bien ete rempli--

28 /\* tous ouvrage parus apres 1990\*/

select \*

from Ouvrage

where Annee > 1990 ;

29/\* insertion de données dans la table Revue\*/

insert into Revue(Nom\_Revue,IdEditeur,Annee\_creation)

values ('Computer Vision, Graphics, and Image Processing',1,1989),

('Information Processing Letters', 2, 1979),

('Transactions on Pattern intelligence', 1, 1965),

('Pattern Recognition', 3, 1965),

('Photogrammetric Engineering and Remote Sensing', 4, 1988),

('Transactions on Systems, Man, and Cybernetics', 2, 1962),

('Proc. of Geovision Applications in Geology', 3, 1984),

('Physical Review', 2, 1985),

('Multimedia Tools and Applications', 4, 1984);

select \*

from Revue -- verification

30 /\* toute les revus qui contient le mot "pattern" \*/

select \*

from Revue

where Nom\_Revue like '%pattern%'

order by IdRevue

31-- insertion de données dans la table Auteur--

INSERT INTO auteur

VALUES ('Abmayr W', 'Abmayr@cs.fhm.edu', 'www.informatik.fh/~abmayr/abmayre.htm'),

( 'Aho A', 'aho@cs.columbia.edu', 'www1.cs.columbia.edu/~aho/'),

('Hopcroft J', 'jeh@cs.cornell.edu', 'www.cs.cornell.edu/jeh/'),

( 'Ullman J', 'ullman@cs.stanford.edu', 'infolab.stanford.edu/~ullman/'),

( 'Akl S', 'akl@cs.queensu.ca', 'www.cs.queensu.ca/~akl/'),

( 'Adams R', null, null),

('Bischof L', 'Leanne.Bischof@csiro.au', 'www.cmis.csiro.au/leanne.bischof/ '),

('Toussaint G', 'godfried@cs.mcgill.ca', 'www-cgrl.cs.mcgill.ca/~godfried/'),

('Powley W', 'wendy@cs.queensu.ca', 'www.cs.queensu.ca/home/wendy/'),

('Amadasun M','amadasun@cs.stanford.edu', 'infolab.stanford.edu/~amadasun/'),

('King R','king@cs.stanford.edu', 'infolab.stanford.edu/~king/')

select \*

from Auteur -- verification

32/\* le nom d'auteur et de site web des auteur ayant un site web\*/

select NomAuteur,SiteWeb

from Auteur

where SiteWeb is not null

33/\* insertion de données dans la table Article\*/

INSERT INTO article (IdArticle,IdRevue,Titre,PageDebut,PageFin,Volume,Serie)

VALUES (7,4, 'graphs random', 121, 129, 64, 2),

( 8,3, 'refinements', 1481, 1491, 39, 8),

( 9,5, 'Seeded region', 641, 647, 16, 6),

( 10,1, 'Radial decom', 325, 332, 55, 5),

(11,6, 'convex algo', 219, 222, 7, 5),

(16,21, 'A Middleware', 17, 19, 10, 2),

(17,22, 'Low-level segm', 216, 218, 21, 1),

(18,23,'Mental workload ', 20,40,1,1)

select \*

from Article -- verification

34/\* le nombre de revues\*/

select

count(idrevue)

from revue

35/\* titre des articles et le nombre de page \*/

select Titre, ( PageFin -PageDebut) as nombre\_page

from Article

------------------------------------------------

36/\* insérer des données dans la table article\_auteur\*/

INSERT INTO article\_auteur(IdArticle,IdAuteur)

VALUES(10, 9),

(7, 6),

(8, 1),

(6, 6),

(17, 14),

(17, 8),

(18, 7),

(16, 11),

(4, 10),

(5, 12),

(11, 5),

(3, 13),

(4, 9),

(1,7);

37/\*selectionner l identifiant de l article ecrit par 'Adams'\*/

select IdArticle

from Article\_Auteur

where IdAuteur in (select IdAuteur

from Auteur

where NomAuteur like '%ADams%' );

38-- deuxieme methode

SELECT IdArticle

FROM Article\_Auteur aa

INNER JOIN (select IdAuteur

from Auteur

where NomAuteur like '%ADams%' ) a ON a.IdAuteur = aa.IdAuteur

39/\*pour chaque article, Son Titre et le Nom de la revue ou il a été publier \*/

select Article.Titre ,Revue.Nom\_Revue

from Article

join Revue on Article.IdRevue=Revue.IdRevue

40 /\* insérer les données dans la table Ouvrage\_Auteur \*/

INSERT INTO ouvrage\_auteur VALUES (5,2),(9,3),(1,2),(4,4)

41/\* le nom des auteurs qui n'ont pas d'ouvrage \*/

select NomAuteur

from auteur

left join Ouvrage\_Auteur on auteur.IdAuteur=Ouvrage\_Auteur.IdAuteur

where IdOuvrage is null;

42--2eme methode

select NomAuteur from Auteur

where IdAuteur in (

select IdAuteur from Auteur

except

select IdAuteur from Ouvrage\_Auteur

);

43/\* insertion de données dans la table motcles \*/

INSERT INTO motcles VALUES (1, 'pattern recognition', null),

(2, 'segmentation', null),

(3, 'graph theory', null),

(4, 'image processing', null),

(5, 'hull', null),

(6, 'broadband Networks', null);

44--select \*

from Article --verification

45/\* insertion de données dans la table article\_motcles \*/

INSERT INTO article\_motcles VALUES (2, 1),

(17, 2),

(4, 4),

(17, 4),

(9, 5),

(10, 6),

(18, 3),

(8, 1),

(3, 2),

(5, 3),

(1, 4),

(11, 1),

(7, 4);

46/\* lister des articl publier dans la revue deep\*/

select \*

from Article

join Revue on Article.IdRevue=Revue.IdRevue

where Nom\_Revue ='deep'

47/\* insertion de données dans la table editeur\*/

INSERT INTO editeur VALUES ('Teubner', '611 7878-285','Germany'),

('Addison-Wesley', '415 402-2500', 'United States'),

( 'elsevier', '20 7611 4500', 'United Kingdom'),

( 'asprs', '01 8038 6860','United Kingdom'),

( 'ieee-smc', '202 371-0101','United States')

48---extract (year from la date systeme)

---- a revoir

select Nom

from Editeur

join Ouvrage on Editeur.IdEditeur=Ouvrage.IdEditeur

where year (getdate()) <

/\*

select \*

from Article

join Revue on Article.IdRevue=Revue.IdRevue

where Article.IdRevue is not null \*/

SELECT DISTINCT E.nom

FROM Editeur E,Revue R, Ouvrage O

WHERE E.idEditeur = R.idEditeur AND E.idEditeur = O.idEditeur;

----------------------------------------------------------------------

50--------------------------------------------------------------------------

select R.IdRevue as IDrevue ,

min(R.annee\_creation) as annee

from revue R

group by (r.IdRevue)

order by annee

51/\*selectionner les nom d articles et le nom de la revue publier dans la plus ancienne revus\*/

SELECT TOP 1 A.titre, R.Nom\_Revue, R.Annee\_creation

FROM ARTICLE A, REVUE R

WHERE A.idRevue = R.idRevue

ORDER BY Annee\_creation ASC;

52--les paires des auteurs qui ont ecrit le meme article

SELECT DISTINCT A1.nomAuteur,A2.nomAuteur

FROM AUTEUR A1, AUTEUR A2, ARTICLE\_AUTEUR AA, ARTICLE AR

WHERE A1.idAuteur = AA.idAuteur AND AA.idArticle = AR.idArticle

AND A1.idAuteur > A2.idAuteur

GROUP BY A1.nomAuteur,A2.nomAuteur

HAVING COUNT(AR.idArticle) = 2

--53==================================================

SELECT distinct ARTICLE\_AUTEUR1.idAuteur,ARTICLE\_AUTEUR2.idAuteur,ARTICLE\_AUTEUR1.idArticle

FROM ARTICLE\_AUTEUR as ARTICLE\_AUTEUR1

join ARTICLE\_AUTEUR as ARTICLE\_AUTEUR2

on ARTICLE\_AUTEUR1.idauteur<>ARTICLE\_AUTEUR2.idauteur

and ARTICLE\_AUTEUR1.idarticle=ARTICLE\_AUTEUR2.idarticle

where ARTICLE\_AUTEUR1.idauteur<ARTICLE\_AUTEUR2.idauteur

;

54/\* le nombre d'ouvrage par auteur \*/

select NomAuteur , count(ouvrage.idouvrage) nombre\_ouvrage

from Ouvrage ,Auteur

group by (auteur.NomAuteur)

55/\*id auteur et le nom de chaque auteur et le nombre d'organismes auxquel il est rattaché\*/

select a.NomAuteur ,COUNT(o.IdOrganisme) nombre\_organisme

from Auteur A ,Organisme O

join Auteur\_Organisme OA on OA.IdOrganisme=o.IdOrganisme

group by (a.NomAuteur)

56--------------18-----

SELECT A.idAuteur, A.nomAuteur, COUNT(AO.idOrganisme) NbreOrganisme

FROM AUTEUR A, AUTEUR\_ORGANISME AO

WHERE A.idAuteur = AO.idAuteur

GROUP BY A.idAuteur, A.nomAuteur;

57--------------------18--------------------f

SELECT AUTEUR\_ORGANISME.idAuteur,COUNT(DISTINCT idOrganisme),nomAuteur

FROM AUTEUR\_ORGANISME

JOIN AUTEUR ON AUTEUR.idAuteur=AUTEUR\_ORGANISME.idAuteur

GROUP BY AUTEUR\_ORGANISME.idAuteur,nomAuteur

ORDER BY AUTEUR\_ORGANISME.idAuteur

58/\* lister le jour le mois et l'année de la date aujourd'hui\*/

SELECT DAY (GETDATE()) Jour, MONTH (GETDATE()) Mois, YEAR (GETDATE()) Année;

59/\* les titre d'article et d'ouvrage dans une seule table \*/

select titre from Article

union

select Titre from Ouvrage

60/\* le nom des revue et le nom des editeur \*/

select coalesce(r.nom\_revue, 'pas de revue') Nom\_Revue , r.IdEditeur , e.Nom

from Revue R

right join Editeur E on r.IdEditeur=e.IdEditeur

61/\* le nom des auteur qui ont ecrit des article et des ouvrages \*/

select distinct au.NomAuteur,aa.IdArticle,oa.IdOuvrage

from Auteur Au

join Article\_Auteur AA on au.IdAuteur=AA.IdAuteur

join Ouvrage\_Auteur OA on OA.IdAuteur=Au.IdAuteur

62/\* les revues qui ont la même année de la première publication et le même éditeur qu'une autre revue \*/

select R1.Nom\_Revue, R2.Nom\_Revue

from Revue R1

join Revue R2 on R1.IdRevue<>R2.IdRevue

where R1.IdEditeur =R2.IdEditeur and R2.Annee\_creation=R1.Annee\_creation

63--------------25-----------------------------------------------

insert into revue

values( 'l''infrmatique facile',8,1996)

64/\*Les revues plus récentes qu’une des revues publiées par l’éditeur ‘Addison- Wesley’ ANY et ALL \*/

--ANY

select \*

from Revue

where Annee\_creation > any (select r.Annee\_creation

from Revue R

join Editeur E on E.IdEditeur=r.IdEditeur

where e.Nom = 'Addison-Wesley')

65 --ALL

select \*

from Revue

where Annee\_creation > all (select r.Annee\_creation

from Revue R

join Editeur E on E.IdEditeur=r.IdEditeur

where e.Nom = 'Addison-Wesley')

67/\*Le nombre d’articles publiés pour chaque membre d'un organisme Canadien\*/

SELECT DISTINCT pays,AUTEUR.idAuteur,COUNT(DISTINCT idArticle) AS NbrArticlePub

FROM ARTICLE\_AUTEUR

JOIN AUTEUR ON ARTICLE\_AUTEUR.idAuteur=AUTEUR.idAuteur JOIN AUTEUR\_ORGANISME ON AUTEUR\_ORGANISME.idAuteur=AUTEUR.idAuteur JOIN ORGANISME ON ORGANISME.idOrganisme=AUTEUR\_ORGANISME.idOrganisme WHERE pays='CANADA'

GROUP BY pays, AUTEUR.idAuteur

68/\*Les ouvrages qui ont plus de deux auteurs \*/

SELECT idOuvrage,COUNT(DISTINCT idAuteur)

FROM OUVRAGE\_AUTEUR

GROUP BY idOuvrage

HAVING COUNT(DISTINCT idAuteur)>2

ORDER BY idOuvrage

69/\*le nombre d'article ecrit par chaque organisme \*/

select idOrganisme,count(distinct idarticle) NbrArticleAOrg

from ARTICLE\_AUTEUR AA

join AUTEUR\_ORGANISME AOon AO.idAuteur=AA.idAuteur

group by idOrganisme

69---

# Table partitionner

# With//translate

# Forcer loptimiseur hint

# Greatest//pivot//

# rollup