Fiches SQL :

I) Introduction aux bases de données :

- Une base de données est un ensemble structuré d'informations enregistré sur un support persistant
- Un SGBD est un logiciel qui gère des bases de données
- Les applications accèdent seulement au niveau logique qui est indépendant du niveau physique (stockage)
- SQL est un langage non procédural qui interagit avec le niveau logique d'une base de données relationnelle

II) Modèle relationnel :

- Une ligne (table EMP) contient toutes les données concernant un employé
- La clé primaire permet de trouver une seule ligne
- Les valeurs de clés primaires sont uniques
- Dans une table, l'ordre des lignes n'a pas d'importance. Ce sont les requêtes qui préciseront l'ordre de tri des résultats
- Un champ peut contenir une valeur nulle (NULL)
- Une valeur nulle représente une information inconnue ou inapplicable

NULL est différent de 0

- La clé primaire (PRIMARY KEY) d'une table est
  le groupe minimal de colonnes qui permet d'identifier une seule ligne
- La colonne clé primaire est sans doublons
- Dans une table, on ne pourra pas insérer 2 lignes avec la même valeur de clé primaire
- La clé primaire est toujours renseignée (NOT NULL)
- Une clé étrangère (FOREIGN KEY) fait référence
  à une clé primaire dans une autre table
- *Les clés étrangères garantissent l'intégrité référentielle*

III) Bases SQL :

Select indique **quelles colonnes** sont sélectionnées et **from** indique depuis **quelles tables.**
La **multiplication** et la **division** sont **prioritaires** sur l'**addition** et la **soustraction.**

*Les alias sont **utiles pour les expressions :***
Ex :
Select ENAME as Nom, SAL*12 as "Salaire Annuel"

from EMP;

Eliminer les lignes en double

```
select distinct DEPTNO
from EMP;
```

IV) Restreindre les lignes :

- Quand la clause **where** est **omise**, une requête renvoie **toutes les lignes de la table**
- **where** agit comme **un filtre.** Seules les lignes qui vérifient la condition sont renvoyées
- **where** suit toujours **from**

```
Select ENAME, DEPTNO
from EMP
where DEPTNO = 10 ;
```

- Les **chaines de caractère** et les **dates** doivent être placées entre **simples quotes**
- Les **valeurs** sont **sensibles à la casse**
- Par défaut, le format des dates est : **YY-MM-DD**

Select ENAME, DEPTNO, JOB
from EMP
where JOB='CLERK' ;

Opérateurs de comparaison

= Egal, **>** Supérieur, **<** Inférieur, **>=** Supérieur ou égal, **<=** Inférieur ou égal,

<> Différent

Select ENAME, SAL, COMM

from EMP

where SAL <= COMM ;

BETWEEN :

Select ENAME, SAL

from EMP

where SAL between 1000 and 1500 ;

LIKE :

Caractères jokers :

_ Remplace **un seul caractère**

% Remplace **0 ou N caractères**

Select ENAME

from EMP

where ENAME like '_A%' ;

IN – Appartenance à une liste :

Select EMPNO, ENAME, SAL, MGR

from EMP

where MGR in (7902,7566, 7788);

Pour tester si un champ vaut NULL :

```
Select ENAME, MGR
from EMP
where MGR is null ;
```

Opérateurs logiques :

Pour combiner **plusieurs conditions :**

AND	True uniquement si les 2 opérandes sont True
OR	True si un des 2 opérandes est True
NOT	True si la condition vaut False

Règles de priorité :

1	Tous les opérateurs de comparaison
2	NOT
3	AND
4	OR

Select EMPNO, ENAME, JOB, SAL

from EMP

where SAL>=1100 and JOB='CLERK';

Vrai si les **2 conditions** sont **vraies**

Select EMPNO, ENAME, JOB, SAL

from EMP

where SAL>=1100 or JOB='CLERK';

Vrai si une des conditions est **vrai**

Select EMPNO, JOB

from EMP

where JOB not in ('CLERK', 'MANAGER', 'ANALYST');

Les parenthèses forcent la priorité

V) Trier les résultats :

ORDER BY

Dans **une table**, **l'ordre des lignes** est **sans importance.** C'est **aux requêtes** qu'il revient de **trier leurs résultats** pour répondre aux **besoins des utilisateurs.**

Select ENAME, JOB, DEPTNO

from EMP

where DEPTNO in (10,20)

order by ENAME;

order by permet de trier les lignes

order by toujours en dernier

dans un select

Par défaut, les lignes sont triées en **ordre croissant** [**asc**]

desc indique que les lignes seront triées en ordre décroissant

```
Select ENAME,
      date_format(HIREDATE, '%y-%m-%d') as HIREDATE
from EMP
      order by HIREDATE desc;
```

Trier sur plusieurs critères :

Select DEPTNO, ENAME

from EMP

order by DEPTNO, ENAME;

Un second critère est utile quand il y a des **valeurs en double sur le premier**

VI) Afficher les données de plusieurs tables :

Ecrire une équijointure :

Select EMP.EMPNO, EMP.ENAME, EMP.DEPTNO, DEPT.DEPTNO, DEPT.LOC

from EMP join DEPT on EMP.DEPTNO = DEPT.DEPTNO ;

Pour simplifier l'écriture des requêtes, utiliser un alias pour désigner chaque table
Select E.EMPNO, E.ENAME, E.DEPTNO, D.DEPTNO, D.LOC

from EMP E join DEPT D on E.DEPTNO = D.DEPTNO ;

Joindre plus de 2 tables :

Select *

from CUSTOMER C join ORD O on C.CUSTID=O.ORDID

join ITEM I on O.ORDID = I.ITEMID ;

Ecrire une non équijointure :

Select E.ENAME, E.SAL, S.GRADE

from EMP E join SALGRADE S

on E.SAL between s.LOSAL and S.HISAL

order by ENAME;

Jointure externe :

Une jointure externe montre aussi les lignes qui ne satisfont pas à la condition de jointure
Select ENAME, DEPT.DEPTNO, DNAME

from EMP right join DEPT on EMP.DEPTNO = DEPT.DEPTNO

order by DEPT.DEPTNO;

Toutes les lignes de la table DEPT à droite de l'opérateur join sont renvoyées

On obtiendrait le même résultat avec :

Select ENAME, DEPT.DEPTNO, DNAME

from DEPT left join EMP on EMP.DEPTNO = DEPT.DEPTNO

order by DEPT.DEPTNO;

Toutes les lignes de la table DEPT à gauche de l'opérateur join sont renvoyées

Joindre une table à elle-même :

Select WORKER.ENAME as Employé, MANAGER.ENAME as Manager

from EMP WORKER join EMP MANAGER on WORKER.MGR = MANAGER.EMPNO

order by WORKER.EMPNO;

La table **EMP** est utilisée **2 fois.** Ici les alias sont nécessaires et donnent du sens

VII) Agrégats et fonctions de groupe :

Moyenne, Somme, Min et Max :

Select avg(SAL), sum(SAL), max(SAL), min(SAL)

from EMP

where JOB like 'SALES%';

Count :

Select count(*), count(job), count(comm), count(distinct job)

from EMP ;

count(*)	Nombre de lignes sélectionnées
count(*colonne*)	Nombre de valeurs non nulles dans la colonne
Count(distinct *colonne*)	Elimine les doublons avant de compter

Valeurs nulles :

Les fonctions de groupe ignorent les valeurs nulles de la colonne.

Select avg(COMM)

from EMP ;

Quel est la moyenne des salaires de chaque département ?

Select DEPTNO, avg(SAL)

from EMP

group by DEPTNO;

GROUP BY regroupe les lignes sélectionnées pour former des groupes

Toute colonne sélectionnée qui n'est pas une fonction de groupe doit être dans GROUP BY
Select DEPTNO, avg(SAL)

from EMP

group by DEPTNO ;

Inversement, les colonnes citées dans GROUP BY ne doivent pas nécessairement être sélectionnées
Select avg(SAL)

from EMP

group by DEPTNO ;

Grouper sur plusieurs colonnes :

Select DEPTNO, JOB, sum(SAL)

from EMP

group by DEPTNO, JOB ;

Exclure des groupes :

HAVING comment ça marche ?

1. Les lignes sont groupées, les fonctions de groupe appliquées
2. Ensuite, seuls les groupes correspondant à la clause HAVING sont affichés

Select DEPTNO, max(SAL)

from emp

group by DEPTNO

having max(SAL) > 2900 ;

En synthèse :

Select JOB, sum(SAL) as "Somme salaires"

from emp

where job not like 'SALES%'

group by JOB

having sum(SAL) > 5000

order by sum(sal) ;

Les clauses doivent apparaitre dans cet ordre **:**

select	quelles sont les colonnes sélectionnées
from	quelles sont les tables utilisées
where	condition sur les lignes
group by	comment les données sont groupées
having	condition sur les groupes
order by	comment les résultats sont triés

VIII) Manipuler les données :

Ajouter une ligne dans une table :

```
insert into dept (deptno, dname, loc)
values (50, 'DEVELOPMENT', 'DETROIT');
```

- ⇨ Une valeur pour chaque colonne non nulle de la table pour laquelle aucune valeur par défaut n'a été définie
- ⇨ Les colonnes peuvent être citées dans un ordre quelconque
- ⇨ Les valeurs doivent correspondre en nombre et en type
- ⇨ Les valeurs de type caractères ou date doivent être entourées par des simples quotes
- ⇨ Pas de valeur en double pour la clé primaire

Insère plusieurs lignes dans une table ::

insert into DEPT (DEPTNO,DNAME,LOC) values

(10,'ACCOUNTING','NEW YORK'),

(20,'RESEARCH','DALLAS'),

(30,'SALES','CHICAGO'),

(40,'OPERATIONS','BOSTON');

Insérer des valeurs nulles :

- Méthode implicite : omettre la colonne dans la liste des colonnes

```
insert into DEPT (DEPTNO, DNAME)
values (50, 'DEVELOPMENT');
```

- Méthode explicite : utiliser le mot clé NULL

```
insert into DEPT (DEPTNO, DNAME, LOC)
values (50, 'DEVELOPMENT', NULL);
```

Insérer une date :

```
insert into EMP (EMPNO, ENAME, JOB, HIREDATE, SAL, DEPTNO)
values (2296, 'AROMANO', 'SALESMAN', '2022-01-31', 1300, 10);
```

La fonction sysdate() enregistre la date et l'heure système

```
insert into EMP (EMPNO, ENAME, JOB, HIREDATE, SAL, DEPTNO)
values (7196, 'GREEN', 'SALESMAN', sysdate(), 2000, 10) ;
```

Modifier des données :

- Utiliser UPDATE pour modifier des lignes existantes

```
update EMP
set DEPTNO = 20
where EMPNO = 7782 ;
```

La clause WHERE indique quelles lignes sont modifiées

Quand la clause WHERE est omise, toutes les lignes de la table sont modifiées :

```
update EMP
set DEPTNO = 20;
```

Utiliser DELETE pour supprimer des lignes

```
delete from DEPT
where DNAME = 'DEVELOPMENT' ;
```

La clause WHERE spécifie quelles lignes sont supprimées

IX) Contrôler les transactions :

- Une transaction est une séquence d'instructions « insert, update, delete » exécutée comme une unité qui peut être validée ou annulée

- Une transaction garantit qu'un groupe logique d'instructions aboutit au complet ou pas du tout

- Les transactions assurent que les données soient toujours dans un état stable et cohérent

start transaction démarre une nouvelle transaction

commit valide la transaction courante et rend les changements permanents

rollback annule les changements en suspens

Etat des données après commit :

- commit rend les changements sur les données permanents dans la base de données
- L'état antérieur des données est définitivement perdu
- Tous les utilisateurs voient le nouvel état des données
- Les verrous sont levés
- Les lignes concernées peuvent être modifiées par un autre utilisateur

start transaction ;

update ...

update ...

insert ...

commit ;

Etat des données après rollback :

- rollback annule tous les changements en attente de validation
- Les changements sont défaits
- L'état antérieur des données est retrouvé
- Les verrous sont levés
- Les lignes concernées peuvent être modifiées par un autre utilisateur

start transaction ;

update ...

update ...

insert ...

rollback ;

SET AUTOCOMMIT	Active/désactive la gestion des transactions
START TRANSACTION	Initie une nouvelle transaction
COMMIT	Valide les changements en attente de validation
ROLLBACK	Annule les changements en cours

X) Utiliser des sous-requêtes :

Utiliser une sous-requête :

select ENAME

from EMP

where SAL > (select SAL

from EMP

where EMPNO = 7566) ;

1. La sous-requête est exécutée une fois avant la requête principale
2. Le résultat de la sous-requête est utilisé par la requête principale
3. La sous-requête est placée entre parenthèses

Opérateurs avec une seule valeur

- Trouver les employés ayant pour manager KING
- Si une sous-requête renvoie une seule valeur
  ⇨ utiliser un opérateur de comparaison
  = > < >= <+ <>

select ename

from emp e

where mgr = (select empno from emp where ename='KING');

Opérateurs avec plusieurs valeurs

- Trouver les départements ayant des employés embauchés en 1982
- Si la sous-requête renvoie plusieurs valeurs
  ⇨ utiliser un opérateur tel que IN ou NOT IN

select *

from dept

where deptno in (select deptno

from emp

where hiredate between '1982-01-01'

and '1982-12-31');

- Les fonctions de groupe ne sont pas autorisées dans la clause where ⇨ utiliser une sous-requête

Mises à jour et sous-requête :

- Augmenter le salaire des employés localisés à DALLAS

update emp

set sal = sal + 1.1

where deptno in (select deptno from dept where loc ='DALLAS');

- Supprimer les départements sans employés

delete from dept
where deptno not in (select deptno from emp) ;

- Toujours évaluer le résultat de la sous-requête pour choisir l'opérateur adéquat
- Si la sous-requête renvoie
  une seule valeur → opérateur de comparaison
- Si la sous-requête peut renvoyer
  une liste de valeurs → IN ou NOT IN
- Pas de ORDER BY dans la sous-requête

XI) Intégrer des fonctions :

Les fonctions dans SQL :

- Manipulent des éléments de données (colonnes, expressions, littéraux)
- Prennent des paramètres en entrée et renvoient un résultat
- Agissent sur chaque ligne sélectionnée
- Peuvent être imbriquées

Fonctions sur les nombres :

- Arrondir

select round(45.926, 0); **46**

select round(45.926, 2); **45.93**

- Tronquer

select truncate(45.926, 0); **45**

select truncate(45.926,2); **45.92**

select sans clause **from** pour tester le comportement d'une fonction avec des données de votre choix.

Fonctions sur les nombres :

- Arrondir le résultat d'un calcul

```
select ENAME, ROUND(SAL*1.0245) as 'Nouveau salaire'
from EMP
```

round est appelé pour chaque ligne sélectionnée

- Arrondir la moyenne des salaires

```
select round(avg(sal)) as Moyenne
from emp
```

Fonctions sur les chaines :

- Concaténer

```
select concat('Hello', 'World');          HelloWorld
select concat('Hello', ' ' , 'World');    Hello World
```

- Modifier la casse

```
select ucase('Hello World') ;             HELLO WORLD
select lcase('Hello World') ;             hello world
```

- Obtenir une sous-chaine

```
select substr('Hello World', 1, 5) ;      Hello
```

- Obtenir la date et l'heure système

```
select curdate();                         2022-05-09
select sysdate();                         2022-05-09 22:19:05
```

- Extraire des éléments d'une date

```
select ename
from emp
where extract(year from hiredate) = 1982 ;
```

Traiter les valeurs nulles :

- **coalesce** remplace NULL par 0 pour le calcul

```
select ename, sal, comm, sal + coalesce(comm, 0) as Total
from emp ;
```

coalesce renvoie la première valeur non nulle de la liste

- Les fonctions peuvent être imbriquées

```
select concat('DW01', substr(ename, 1,4)) as identifiant
from emp
```

https://dev.mysql.com/doc/refman/8.0/en/built-in-function-reference.html