SQL

I) Bases de données et tables :

Un S.G.B.D. est un outil pour organiser, stocker, modifier, calculer et extraire des infos.
Une B.D.R. peut être considérée comme un ensemble de tables à 2 dimensions.

Exemple :

Nom	Prénom	Commissions
Dupond	Jean	12000
Durand	Paul	8000
Martin	Jacques	16000
.....	.....	.....

Les commandes associées à la définition de table sont :
CREATE TABLE (Création)
ALTER TABLE (Modification de la structure)
DROP TABLE (Effacement)

Pour modifier le contenu des tables, on utilise :
INSERT (Ajout)
DELETE (Effacement)
UPDATE (Modification)
ROLLBACK annuler les dernières modifications.
COMMIT valide les modifications.

II) Syntaxe des instructions :

Toutes les instructions SQL doivent se terminer par un point-virgule (;)

Créer une base de donnée :

CREATE DATABASE [chemin d'accès]nom_base ;

Exemple :
```
CREATE TABLE employe
( nomchar(20),
prenom char(15),
age integer ,
statut char(15) ,
salaire integer ) ;
```

Ouvrir une base de donnée :

START DATABASE nom_base ;

Fermer la base de donnée :

STOP DATABASE ;

Effacer une table :

DROP TABLE nom_table ;

III) Modifier base de données :

Insérer colonnes :

ALTER TABLE nom_table ADD (nom_col type,.....) ;

Exemple : Insérer la colonne anc (ancienneté) dans la table employe.
ALTER TABLE employe ADD (anc numeric (5,2)) ;

Insérer lignes :

INSERT INTO nom_table (nom_col_1 , ... , nom_col_n) VALUES (valeur_1 , ... , valeur_n) ;

Exemple : Insérer l'employé Dupont Jean avec le statut cadre dans la table employe.
INSERT INTO employe (nom , prenom , statut) VALUES ('Dupont', 'Jean' , 'cadre') ;

Effacer lignes :

DELETE FROM nom_table WHERE condition de sélection ;

Exemple : Effacer dans la table des employés, toutes les personnes qui sont parties en retraite.
DELETE FROM employe WHERE statut = 'R' ;

Modifier lignes :

UPDATE nom_table SET nom_col = expression ,.....[WHERE condition de recherche] ;

Exemple : L'employé Dupont aura son salaire fixé à 13000 F.
UPDATE employe SET salaire = 13000 WHERE nom = ' Dupont ' ;

Tous les salaires sont augmentés de 5 % :
UPDATE employe SET salaire = salaire * 1.05 ;

Seuls les cadres sont augmentés :
UPDATE employe SET salaire = salaire * 1.05 WHERE statut = ' Cadre ' ;

IV) Interroger une table :

Une commande de sélection contient les clauses suivantes :
SELECT ..... FROM ..... WHERE ..... GROUP BY ..... HAVING ..... ORDER BY .....

Sélection toues les colonnes :

SELECT * FROM nom_table ;

Sélection une ou plusieurs colonnes :

SELECT nom_col , ... FROM nom_table ;

Exemple : Afficher le nom, prénom et age des employés.
SELECT nom , prenom , age FROM employe ;

Sélection de lignes :

SELECT nom_col , ...FROM nom_tableWHERE critère ;

Exemple : Afficher le nom, prénom et age des employés qui ont un salaire supérieur à 8 000 et inférieur à 20 000 €.
SELECT nom , prenom , age FROM employe WHERE salaire > 8000 AND salaire < 20000 ;

V) Critères de sélection :

Between :

Valider si une expression est comprise entre 2 valeurs.
Expression1 [NOT] BETWEEN Expression2 AND Expression3

Exemple : Afficher le nom, prénom et age des employés qui ont un salaire compris entre 8 000 et 20 000 €.

SELECT nom, prenom FROM Employe WHERE Salaire BETWEEN 8000 and 20000 ;

Like :

Rechercher des valeurs possédant certaines caractéristiques. On peut utiliser les caractères jokers :
_ (souligné) qui remplace n'importe quel caractère ;
%(pourcentage) qui remplace n'importe quelle chaîne de caractères.

Nom_Col [NOT] LIKE Expression

Exemple : Sélectionner tous les employés dont le nom commence par Du.
SELECT Nom, Prenom FROM Employe WHERE Nom LIKE 'Du%' ;

Exemple : Sélectionner tous les employés dont le nom contient un R en 3ème position.
SELECT Nom, Prenom FROM Employe WHERE Nom LIKE '_ _R%' ;

Exemple : Sélectionner tous les employés dont le nom contient "RA".
SELECT Nom, Prenom FROM Employe WHERE Nom LIKE '%RA%' ;

Exists :

Teste si le résultat d'une sélection contient au moins une ligne.
EXISTS (sélection) ;

Exemple : Tester si au moins un employé à une ancienneté de 0.
SELECT nom FROM employe WHERE EXISTS (SELECT * FROM Employe WHERE anc = 0) ;

IN :

Sélectionne les lignes contenant une valeur coïncidant avec l'une des valeurs dans une liste.

expression [NOT] IN liste ou expression [NOT] IN Sous-sélection

Exemple : Sélectionner tous les employés ayant 64 ou 65 ans.
SELECT * FROM employe WHERE age IN (64 , 65) ;

Opérateurs :

On peut aussi utiliser les opérateurs :
booléens : AND, OR et NOT, de comparaison : = , > , et < (et leur combinaisons)

Exemples : Sélectionner les employés qui n'ont pas le statut directeur.
SELECT * FROM Employe WHERE NOT statut = 'Directeur' ;
ou
SELECT * FROM Employe WHERE statut ! = 'Directeur' ;

ALL :

Utilisé avec un opérateur de comparaison, sert à tester si une expression est vérifiée dans tous les cas de figure. L'expression est juste si la comparaison est vérifiée pour toutes les valeurs renvoyées par la clause "sous-sélection".

Exemple : Vérifier si tout le personnel a moins de 65 ans. Si oui , on affiche " * FROM employe ".
SELECT * FROM employe WHERE 65 > ALL (SELECT age FROM employe) ;

ANY :

Contrairement à ALL, l'expression est juste si la comparaison est vérifiée dans la clause de sous-sélection pour au moins une valeur.

Exemple : si au moins une valeur de la sélection " SELECT age FROM employe" est > à 65, alors on affiche " * FROM employe "
SELECT * FROM employe WHERE 65 < ANY (SELECT age FROM employe) ;

VI) Tri des résultats :

Croissant/décroissant :

Il est possible de trier les résultats d'une sélection avec la clause ORDER BY :
ORDER BY nom_col [ASC / DESC]

Exemple : On sélectionne les employés et on les classe par statut ; pour chaque statut, ils sont classés par salaire, le salaire le moins important en premier.
SELECT statut, nom, prenom, salaire FROM employe ORDER BY statut ASC, salaire ASC ;

Les doublons :

Pour éviter de sélectionner des lignes en double (redondantes), on utilise la clause DISTINCT.
SELECT DISTINCT nom_col, ... FROM nom_table WHERE critère ;

Exemple : Afficher tous les noms des employés qu'une seule fois.
SELECT DISTINCT nom FROM employe

VII) Les groupes de valeurs :

GROUP BY :

Permet de grouper les lignes par type.
GROUP BY nom_col , ...

Exemple : Donner, statut par statut, le total des salaires des employés.
SELECT statut, SUM(salaire) FROM employe GROUP BY statut ;

Exemple : Afficher les statuts et le salaire par statut, comptabiliser le nombre d'employé par statut.
SELECT statut, COUNT(*), SUM(salaire) FROM employe GROUP BY statut ;

Statut	COUNT	SUM (salaire)
Directeur	8	86000
Cadre	12	176000
Ouvrier	14	123000

Il y a 12 personnes qui sont cadre et le total de leur salaire se monte à 176000 francs.

HAVING :

Permet de spécifier des critères de sélections sur les groupes de valeur. Accompagne toujours le GROUP BY.
GROUP BY .....HAVING condition

Exemple : Afficher les statuts et le salaire par statut, comptabiliser le nombre d'employé par statut. On ne prend en compte que les statuts qui comprennent plus de 10 personnes.
SELECT statut, COUNT(*), SUM(salaire) FROM employe GROUP BY statut HAVING COUNT (*) > 10 ;

Statut	COUNT	SUM (salaire)
Cadre	12	176000
Ouvrier	14	123000

Remarque : La différence entre HAVING et WHERE est que WHERE s'applique à des lignes seules et que HAVING s'applique à des groupes de lignes.

VIII) Les fonctions numériques :

AVG

Permet de calculer la moyenne de valeurs.
AVG ([ALL] expression) ou AVG (DISTINCT nom_col)

Exemple : Calculer la moyenne des âges des employés.
SELECT AVG (age) FROM employe ;

COUNT :

Comptage du nombre d'élément sélectionnés.
COUNT (*) ou COUNT (DISTINCT nom_col)

Exemple : Compter le nombre de cadres.
SELECT COUNT(*) FROM employe WHERE statut='cadre' ;

Exemple : Compter le nombre de status différents au sein de l'entreprise.
SELECT COUNT (DISTINCT statut) FROM employe ;

MAX :

Détermine la valeur maximale d'un ensemble de valeurs.
MAX ([ALL] expression) ou MAX (DISTINCT nom_col)

Exemple : Trouver l'âge le plus élevé dans la catégorie cadre.
SELECT MAX (age) FROM employe WHERE statut='cadre' ;

MIN :

Semblable à MAX, pour le minimum de valeurs.

SUM :

Permet de déterminer la somme de valeurs.
SUM ([ALL] expression) ou SUM (DISTINCT nom_col)

Exemple : Trouver le total des salaires des cadres.
SELECT SUM (salaire) FROM employe WHERE statut='cadre' ;

IX) Interrogation de tables multiples

Interrogation sur plusieurs tables :

Il est possible d'utiliser plus d'une table dans une instruction " SELECT .... FROM ...."
SELECT nom_col1, nom_col2, ... FROM table1, table2 ...WHERE .....

Exemple :
SELECT nom, CA , SUM(facture) FROM Ventes, Compta
WHERE Ventes.nom = Compta.nom AND SUM (facture) > 10000 GROUP BY nom;

L'exemple sélectionne les ingénieurs commerciaux dont le total de facture en cours de leurs clients dépasse 10000 francs. On accède ici à 2 tables qui sont ventes et compta.

Le critère qui permet la jointure des deux tables est : Ventes.nom = Compta.nom. Il indique que pour un même ingénieur commercial, on va chercher dans la table Vente le chiffre d'affaire (CA) et dans la table Compta, la somme des factures en cours.

Sous requêtes :

Il est possible de mettre des SELECT à l'intérieur d'autres SELECT : on peut alors faire des requêtes à partir de résultats d'autres requêtes (requêtes imbriquées).

Les sous-requêtes sont utilisées avec les opérateurs : IN, ALL, ANY, EXISTS, SOME.

X) Opérations sur les ensembles :

Union :

L'union de 2 tables produit une table logique qui contient à la fois les lignes renvoyées par la première sélection et les lignes de la seconde sélection.

sélection1 UNION [ALL] sélection2 [ORDER BY nom_col [ASC / DESC]] ;

Exemple : Obtenir une table composée des clients et des fournisseurs.
Les clients qui sont également des fournisseurs ne figureront qu'une seule fois dans la table résultante.
SELECT Societe, cp FROM Client UNION SELECT Societe, codepost FROM fournis ;

Intersection :

Sur certaines versions de SGBD SQL, il est possible d'effectuer des intersections de tables.

sélection1 INTERSECT sélection2 [ORDER BY nom_col [ASC / DESC]] ;

Exemple : Trouver les clients qui sont également fournisseurs.
SELECT societe, cp FROM client INTERSECT SELECT societe, codepost FROM fournis ;

Exclusion :

On utilise l'opérateur MINUS (qui n'existe que dans certaines versions de SGBD).

sélection1 MINUS sélection2 [ORDER BY nom_col [ASC / DESC]] ;

Les jointures :

La jointure est une opération permettant de combiner des informations venant de plusieurs tables.

Les différents types de jointures sont :
L'équi-Jointure :
Elle permet de rapprocher (de juxtaposer) des informations issues de deux tables distinctes qui ont un attribut commun (dont les valeurs sont prises dans le même domaine) appelé attribut de jointure.

Jointure d'une table à elle-même :
Il peut être utile de rassembler des informations venant d'une ligne d'une table avec des informations venant d'une autre ligne de la même table.

Les sauvegardes :

Sauvegarde à froid :
Le serveur SQL est arrêté, tous les clients sont déconnectés, on effectue la sauvegarde, puis on relance le serveur SQL.

Sauvegarde à chaud :
La sauvegarde est effectuée pendant que les utilisateurs sont connectés à la base de données