

Mémo des fonctions SQL

I. La sélection des données :

SQL	Opération	Syntaxe	Exemple
SELECT	Sélectionner toutes les colonnes de la table	SELECT * FROM table;	Afficher tout le contenu de la table stagiaires : SELECT * FROM stagiaires;
	Sélectionner seulement certaines colonnes	SELECT colonne1, colonne2, FROM table;	Afficher le nom et le prénom des stagiaires: SELECT nom, prenom FROM stagiaires;
DISTINCT	Eliminer les doublons	SELECT DISTINCT colonne1 FROM table;	afficher les différents code postaux des professeurs : SELECT DISTINCT code FROM professeurs;
WHERE	restreindre le nombre de lignes renvoyées avec une condition	SELECT * FROM table WHERE condition1 AND condition2 ...;	afficher la liste des stagiaires qui habitent Nice: SELECT * FROM stagiaires WHERE ville ='Nice';
Les opérateurs de comparaison	Les conditions sont posées grâce aux opérateurs suivants: = égal != différent < inférieur > supérieur <= inférieur ou égal >= supérieur ou égal	SELECT * FROM table WHERE colonne1 <i>opérateur</i> colonne2; SELECT * FROM table WHERE colonne1 opérateur valeur;	Afficher les évaluations dont la note est supérieur à 15: SELECT * FROM evaluations WHERE note > 15;
BETWEEN	Définir un intervalle d'application	SELECT * FROM table WHERE colonne BETWEEN expression1 AND expression2;	Afficher les évaluations dont la note est comprise entre 10 et 15: SELECT * FROM évaluations WHERE note BETWEEN 10 AND 15;
IN	Chercher une valeur dans une liste de choix.	SELECT * FROM table WHERE colonne IN (expression1,expression2,...);	Afficher les évaluations dont la note est comprise entre 10 et 15: SELECT * FROM évaluation WHERE note IN (10, 11, 12, 13);
LIKE	Comparer les chaînes de caractères où chaîne peut contenir des caractères jokers: _ remplace 1 caractère % remplace une chaîne de caractère de longueur quelconque.	SELECT * FROM table WHERE colonne LIKE 'chaîne' ; SELECT * FROM table WHERE colonne LIKE 'ch_in%' ;	Afficher la liste des stagiaires qui ont un nom commençant par "Du" et dont le code commence par 06 et finit par 00. SELECT * FROM stagiaires WHERE nom LIKE 'Du%' AND code LIKE '06_00';
NULL	Savoir si le champ a été saisi ou non	SELECT * FROM table WHERE colonne1 IS NULL AND colonne2 IS NOT NULL ;	Afficher la liste des stagiaires dont le nom est saisi mais pas encore l'adresse : SELECT * FROM stagiaires WHERE nom IS NOT NULL AND adresse IS NULL ;

II. Ordonner les valeurs

SQL	Opération	Syntaxe	Exemple
La clause ORDER BY	permet de classer le résultat d'un select par ordre croissant (ASC) ou décroissant (DESC)	SELECT colonne1,colonne2,... FROM table ORDER BY colonne1 ; (ordre croissant par défaut) SELECT colonne1,colonne2,... FROM nom_table ORDER BY colonne1, colonne2 DESC ;	Faire une liste des élèves par ordre alphabétique: SELECT nom, prenom FROM stagiaires ORDER BY nom, prenom ASC ; Faire une liste des matières par coefficients croissants: SELECT titre, coeff FROM matieres ORDER BY coeff;
AS	Permet de renommer une en-tête de colonne du résultat. Ce mot clef est très pratique pour donner un nom de colonne pour un résultat de calcul.	SELECT colonne1, colonne2 AS nouveau_nom_colonne FROM nom_table;	Afficher les prix horaires nets, sachant qu'il y a 10% de différence entre la valeur brute et la valeur nette: SELECT ROUND ((prix_horaire*110/100),2) AS "Prix Nets" FROM remuneration;

III. Opérateurs et fonctions

SQL	Opération	Syntaxe	Exemple
Les opérateurs logiques	AND : ET logique où la condition1 et la condition2 doivent être vérifiées OR : OU logique où il suffit que l'une des conditions au moins soit vérifiée NOT : NON logique qui permet de tester le contraire d'une condition	SELECT * FROM table WHERE condition1 AND condition2; SELECT * FROM table WHERE condition1 OR condition2; SELECT * FROM nom_table WHERE colonne1 NOT IN (expression1, expression2) ;	Afficher la liste des stagiaires dont le nom est saisi mais pas encore l'adresse ou qui ont un identifiant >150: SELECT * FROM stagiaires WHERE (nom IS NOT NULL AND adresse IS NULL) OR id_stagiaire > 150;
Fonctions et opérateurs arithmétiques	Effectuer des calculs sur des attributs + addition - soustraction * multiplication / division ABS (val) valeur absolue de val ROUND (val, nb) arrondi de val à nb chiffres après la virgule TRUNC (val, nb) renvoie val tronqué à nb chiffres après la virgule		Ramener les notes sur 10 au lieu de 20 et les arrondir à un chiffre après la virgule: SELECT ROUND (note / 2, 2) FROM evaluations;

Fonctions et opérateurs sur les chaînes de caractères	<p>chaîne1 chaîne2 : concatène les deux chaînes CONCAT (chaîne1, chaîne2)</p> <p>INITCAP (chaîne) met le premier caractère de chaque mot de chaîne en majuscule et le reste du mot en minuscules.</p>	<p>SELECT colonne1 '=>' colonne2 FROM table; SELECT CONCAT(colonne1, colonne2) FROM table; SELECT INITCAP(nom_colonne) FROM table;</p>	<p>Afficher la liste des noms et prenom des stagiaires séparés par un « . » SELECT nom '.' prenom FROM stagiaires;</p> <p>Afficher la liste des noms et prenom des stagiaires collés l'un à l'autre. SELECT CONCAT (nom, prenom) FROM stagiaires;</p>
	<p>UPPER (chaîne) met tous les caractères de la chaîne en lettres majuscules. LOWER (chaîne) met tous les caractères de la chaîne en lettres minuscules. LENGTH(chaîne) renvoie la longueur de chaîne. REPLACE (nom_champs, ancienne_chaine, nouvelle_chaine) remplace dans le champ nom_champs, les occurrences d'ancienne_chaine par nouvelle_chaine.</p>	<p>SELECT UPPER(nom_colonne) FROM table; SELECT LOWER(nom_colonne) FROM table; SELECT LENGTH(colonne) FROM table; SELECT REPLACE (colonne, 'ancienne_chaine', 'nouvelle_chaine') FROM stagiaires;</p>	<p>Afficher la liste des noms des stagiaires avec la première lettre en majuscule. SELECT INITCAP(nom) FROM stagiaires;</p> <p>Afficher les noms des stagiaires en majuscules. SELECT UPPER(nom) FROM stagiaires;</p> <p>Afficher les noms des stagiaires en minuscules. SELECT LOWER(nom) FROM stagiaires;</p> <p>Afficher la longueur des noms des stagiaires. SELECT LENGTH(nom) FROM stagiaires; SELECT REPLACE (nom, 'Dupont', 'Durand') FROM stagiaires;</p>
Fonctions et opérateurs sur les dates	<p>=, !=, <, >, <=, >= permettent de comparer deux dates entre elles</p>	<p>SELECT * FROM table WHERE colonne_date < date; SELECT date2 - date1 FROM table;</p>	<p>Lister les stagiaires dont la date de naissance est postérieure au 01/01/1980: SELECT *FROM stagiaires WHERE datenaiss > '1980-01-01';</p> <p>Afficher le nom et l'âge des stagiaires au 1er mars 2003: SELECT nom, ('2003-03-01' - datenaiss)/365 AS "Age" FROM stagiaires;</p>

IV. Grouper les données

SQL	Opération	Syntaxe	Exemple
GROUP BY	Permet de subdiviser une table en plusieurs groupes. Chaque groupe regroupe l'ensemble des lignes ayant la même valeur pour le champ spécifié	SELECT colonne1, count(colonne2) FROM nom_table GROUP BY nom_champs1;	Comptez le nombre d'étudiant par option: SELECT option, count(*) FROM stagiaires GROUP BY option;
HAVING	Permet de préciser les conditions dans lesquelles le GROUP BY doit s'exécuter. Elle fonctionne de la même façon que la clause WHERE .	SELECT colonne1, count(colonne2) FROM table GROUP BY colonne1, colonne2, ...; HAVING condition;	Comptez le nombre d'étudiant pour les options Bureautique et Infographie: SELECT option, count(*) FROM stagiaires GROUP BY option HAVING option in ('Bureautique', 'Infographie');
Les Fonctions de groupe	AVG : calcule la moyenne des valeurs contenues dans champs_numérique COUNT : compte le nombre de lignes du résultat de la requête MIN : sélectionne la plus petite valeur contenu dans le champ nom_champs MAX : sélectionne la plus grande valeur contenu dans le champs nom_champs SUM : calcule la somme des valeurs de nom_champs STDDEV : calcule l'écart-type des valeurs de nom_champ VARIANCE : calcule la variance des valeurs de nom_champs	SELECT AVG (champs_numérique) FROM nom_table; SELECT COUNT (nom_colonne1*) FROM nom_table; SELECT MIN(nom_champs) FROM nom_table; SELECT MAX(nom_champs) FROM nom_table; SELECT SUM(nom_champs) FROM nom_table; SELECT STDDEV(nom_champs) FROM table; SELECT VARIANCE (nom_champs) FROM table;	Afficher la moyenne des notes des évaluations SELECT AVG(note) FROM évaluations; Compter le nombre de stagiaires: SELECT COUNT(*) FROM stagiaires; Afficher le coefficient minimum des matières: SELECT MIN(coefficient) FROM matieres; Afficher le coefficient maximum des matières: SELECT MAX(coefficient) FROM matieres; Calculer la somme des coefficients des matières: SELECT SUM(coefficient) FROM matieres; Calculer l'écart-type des notes: SELECT STDDEV(note) FROM evaluations; Calculer la variance des notes: SELECT VARIANCE(note) FROM evaluations;

V. Les jointures

SQL	Opération	Syntaxe	Exemple
INNER JOIN	Jointure interne (INNER JOIN): permet de regrouper les données présentes dans deux tables si elles ont un champ commun.	SELECT table1.colonne1, table2.colonne2, ... FROM table1 INNER JOIN table2 USING (colonne) WHERE condition; <=> SELECT table1.colonne1, table2.colonne2, ... FROM table1, table2 WHERE table1.colonne = table2.colonne [AND condition]; Rq: il n'est nécessaire de mettre le nom de la table devant le nom colonne que si les deux tables ont une colonne du même nom. Rq: par défaut la jointure est une jointure interne, il suffit donc de mettre JOIN plutôt qu' INNER JOIN . Rq: si la colonne qui sert de jointure ne porte pas le même nom dans les deux tables, on peut utiliser la syntaxe suivante: SELECT table1.colonne1, table2.colonne2, ... FROM table1 JOIN table2 USING (colonne1 = colonne2) WHERE condition;	Afficher la liste des matières suivies du nom du professeur qui les enseigne: SELECT titre, nom FROM matieres JOIN prof USING (id_prof); <=> SELECT titre, nom FROM matieres, prof WHERE matieres.id_prof=prof.id_prof;
CROSS JOIN	Jointure croisée: permet de faire le produit cartésien de deux tables.	SELECT table1.colonne1, table2.colonne2, ... FROM table1 CROSS JOIN table2 WHERE condition;	Obtenir la liste de tous les couples professeur -matière possibles: SELECT nom, titre FROM prof CROSS JOIN matieres;
LEFT OUTER JOIN	Jointure avec tous les éléments de la table de gauche même ceux qui n'ont pas de correspondance dans la table de droite.	SELECT table1.colonne1, table2.colonne2, ... FROM table1 LEFT OUTER JOIN table2 USING (colonne) WHERE condition;	Lister tous les professeurs suivi de la matière qu'ils enseignent ou d'un blanc s'ils n'enseigne pas pour l'instant. SELECT nom, titre FROM prof LEFT OUTER JOIN matieres USING (id_prof);
RIGHT OUTER JOIN	Jointure avec tous les éléments de la table de droite même ceux qui n'ont pas de correspondance dans la table de gauche.	SELECT table1.colonne1, table2.colonne2, ... FROM table1 RIGHT OUTER JOIN table2 USING (colonne) WHERE condition;	lister les matières avec le nom des professeurs qui les enseignent ainsi que les matières qui sont pour l'instant sans enseignant. SELECT titre, nom FROM prof RIGHT OUTER JOIN matieres USING (id_prof);

FULL OUTER JOIN	Union des deux précédentes jointures: permet d'obtenir tous les éléments des deux tables même lorsqu'il n'y a pas de correspondance.	SELECT table1.colonne1, table2.colonne2, ... FROM table1 FULL OUTER JOIN table2 USING (colonne) WHERE condition;	Faire une liste complète des matières et des professeurs. SELECT titre, nom FROM prof FULL OUTER JOIN matieres USING (id_prof);
JOINTURE PAR INEGALITE	Toutes les requêtes sur les jointures précédentes sont des jointures par égalité entre les deux colonnes qui servent de jointure. Les jointures par inégalité se font grâce à un des opérateurs suivants: <, >, <=, >=, <>, !=, BETWEEN, IN, LIKE	SELECT table1.colonne1, table2.colonne2, ... FROM table1 JOIN table2 USING (colonne1 op colonne2) WHERE condition; <=> SELECT table1.colonne1, table2.colonne2, ... FROM table1, table2 WHERE table1.colonne op table2.colonne [AND condition];	Affichez la liste des professeurs dont l'indice de rémunération est supérieure à celle de STICCA Lucie. SELECT prof.nom, prof.indice FROM prof JOIN prof p USING (prof.indice > p.indice) AND p.nom='STICCA' AND p.prenom='Lucie';
AUTOJOINTURE	C'est la jointure d'une table sur elle-même, elle est utilisée lorsque l'on souhaite comparer un ensemble d'enregistrement à un enregistrement de cette table.	SELECT table.colonne1, t.colonne2, ... FROM table, table t WHERE table.colonne op t.colonne [AND condition];	

VI. Opérateurs ensemblistes

SQL	Opération	Syntaxe	Exemple
UNION	le résultat contiendra le résultat de requête1 + le résultat de requête	requête1 UNION requête2	Afficher la liste de toutes les personnes présentes dans l'établissement: SELECT nom, prenom FROM prof UNION SELECT nom, prenom FROM stagiaires;
INTERSECT	le résultat contiendra les lignes communes aux deux requêtes	requête1 INTERSECT requête2:	Afficher la liste des profs qui sont aussi stagiaires: SELECT nom, prenom FROM prof INTERSECT SELECT nom, prenom FROM stagiaires;
EXCEPT	résultat de requête1 - résultat de requête2	requete1 EXCEPT requête2	Afficher la liste des profs qui ne sont pas stagiaires: SELECT nom, prenom FROM prof EXCEPT SELECT nom, prenom FROM stagiaires;

VII. Insérer des Données

SQL	Opération	Syntaxe	Exemple
INSERT	Ajouter un nouvel enregistrement à une table.	INSERT INTO table (colonne1, colonne2, ...) VALUES (valeur_colonne1, valeur_colonne2, ...); Si l'on ne précise aucune colonne, par défaut la liste des colonnes sera la liste complète des champs de la table. Par contre dans ce cas-là, il faut bien donner une valeur pour tous les champs de la table dans le VALUES . INSERT INTO table VALUES (valeur_colonne1,	Insérer un stagiaire dont on connaît seulement l'identifiant, le nom et le prénom: INSERT INTO stagiaires VALUES (100, 'STICCA','Lucie'); Insérer un stagiaire dont on connaît tous les renseignements: INSERT INTO stagiaires VALUES (100, 'STICCA', 'Lucie', '62, Impasse du 8 Mai 1945','06700', 'Saint Laurent du Var', '04 98 08 44 93', 'F', '1976-07-24', 'Programmation');

VIII. Modifier des Données

SQL	Opération	Syntaxe	Exemple
UPDATE	Modifier le contenu d'un ou plusieurs enregistrements.	UPDATE table SET colonne1 = nouvelle_valeur1, colonne2 = nouvelle_valeur2, ... WHERE condition;	Modifier l'enregistrement de STICCA Lucie qui vient de déménager: UPDATE stagiaires SET adresse=' 32 bd Auguste Gal', code='06300', ville='Nice' WHERE nom='STICCA' AND prenom='Lucie';

IX. Supprimer des Données

SQL	Opération	Syntaxe	Exemple
DELETE	Supprimer une ou plusieurs lignes d'une table. Attention, si aucune condition n'est spécifiée, le contenu entier de la table sera effacé.	DELETE FROM table WHERE condition;	Effacer la ligne de la table stagiaire concernant STICCA Lucie: DELETE FROM stagiaires WHERE nom='STICCA' et prenom='Lucie'; Effacer tout le contenu de la table stagiaire: DELETE FROM stagiaire

X. Requêtes Imbriquées

SQL	Opération	Syntaxe	Exemple
Sous-requête renvoyant une seule valeur	Le résultat de la sous-requête est considéré comme une constante.	SELECT colonne1, colonne2 FROM table1 WHERE colonne = (SELECT colonne3 FROM table2 WHERE condition);	Afficher la liste des profs qui ont un indice égal à celui de STICCA Lucie: SELECT nom, indice FROM prof WHERE indice = (SELECT indice FROM prof WHERE nom='STICCA' AND prenom='Lucie');
Sous-requête renvoyant un ensemble de valeurs	Permet de comparer la valeur d'une colonne avec l'ensemble de valeurs renvoyé par la sous-requête. Cette comparaison se fait avec les opérateurs IN , ANY ou ALL .	SELECT colonne1, colonne2 FROM table1 WHERE colonne IN (SELECT colonne3 FROM table2 WHERE condition);	Comptez le nombre d'évaluations effectuées par des stagiaires qui sont des filles: SELECT COUNT(*) FROM evaluations WHERE id_stagiaire IN (SELECT id_stagiaire FROM stagiaires WHERE sexe='F');
Sous-requête renvoyant plusieurs colonnes	Permet de comparer simultanément plusieurs colonnes avec le résultat de la sous-requête.	SELECT colonne1, colonne2 FROM table1 WHERE colonne1, colonne2 = (SELECT colonne3,colonne4 FROM table2 WHERE condition)	Chercher s'il y a un prof qui est aussi stagiaire: SELECT nom, prenom FROM prof WHERE (nom, prenom) = (SELECT nom, prenom FROM stagiaires);
Sous-requête existentielle	Affiche le résultat de la première requête si la colonne existe dans la seconde requête. On utilise les mots clefs: EXISTS et NOT EXISTS	SELECT colonne1, colonne2 FROM table1 WHERE colonne EXISTS (SELECT colonne3 FROM table2 WHERE condition);	Affichez les noms et prénoms des profs qui ont le meilleur indice: SELECT nom, prenom, indice FROM prof p1 WHERE NOT EXISTS (SELECT * FROM prof p2 WHERE p2.indice >p1.indice);