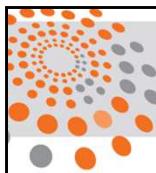


Interroger des bases de données avec le langage SQL



Ce support est la propriété d'IB SA



Plan

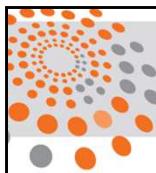
1. Introduction

- Rappels sur le modèle relationnel
- Les caractéristiques du langage SQL

2. Le Langage d'Interrogation de Données (LID)

- La sélection de données
- La gestion des valeurs null
- Les restrictions ou conditions
- Les tris
- Les jointures





Plan

3- Utilisation des fonctions

4- Utilisation des opérateurs ensemblistes

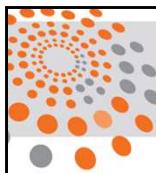
- Union
- Intersect
- Minus ou Except

5. Utilisation des sous interrogations

6. Le Langage de Manipulation de Données (LMD)

- Le Insert
- L'Update
- Le Delete



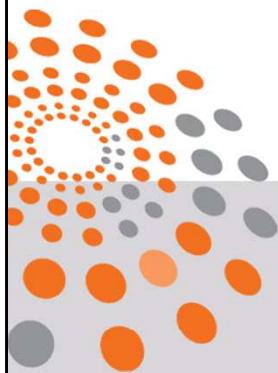


Plan

7. Notions sur le langage de définition de données (LDD)

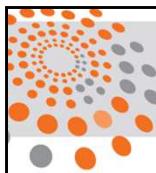
- Création de tables : syntaxe
- Types de données
- Types de contraintes
- Modifier la définition d'une table
- Supprimer une table
- Notions sur : Vue, index et synonyme





Module 1 Introduction



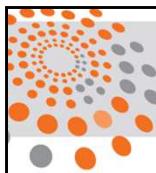


Introduction

Sommaire Module 1

- Rappel sur le modèle relationnel
- Les caractéristiques du langage SQL





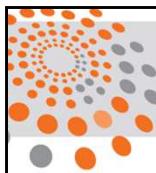
Rappel sur le modèle relationnel

Une base de données est un ensemble cohérent d'informations mémorisées sur support informatique.

Ces informations sont accessibles à l'aide d'une application appelée système de gestion de base de données (SGBD). Si ce SGBD est basé sur le modèle relationnel de CODD, on dit qu'il s'agit d'un système de gestion de base de données relationnel (SGBDR).

Pour dialoguer avec un SGBDR on utilise le langage SQL. Ce langage permet de soumettre des requêtes (des questions) au SGBDR.





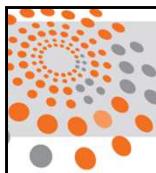
Rappel sur le modèle relationnel

Le modèle relationnel est constitué d'un ensemble d'opérations formelles sur les relations.

Les données sont stockées dans des tables qu'on peut mettre en relation.

Une relation se fait entre des colonnes ayant des données qui correspondent.





Rappel sur le modèle relationnel

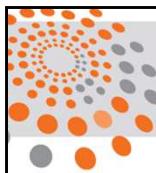
La modélisation relationnelle permet de représenter les relations à l'aide de tables (à deux dimensions)

Une ligne d'une table représente donc une entité.

Un attribut est le nom des colonnes qui constitue la définition d'une table. Il comporte un typage de données.

On appelle tuple (ou n-uplet) une ligne de la table.





Rappel sur le modèle relationnel

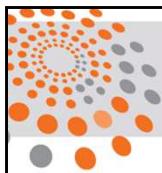
La cardinalité d'une relation est le nombre de tuples qui la composent.

La clé principale (ou primaire) d'une relation est l'attribut, ou l'ensemble d'attributs, permettant de désigner de façon unique un tuple.

Une clé étrangère, par contre, est une clé faisant référence à une clé appartenant à une autre table.

Le nom des colonnes à relier peut être différent, mais le type de données doit être le même.

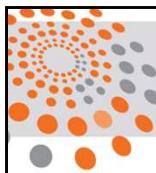




Rappel sur le modèle relationnel

Cas Pratique :
La base de données du stage



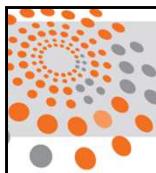


Rappel sur le modèle relationnel

Caractéristiques :

- Indépendance physique : le niveau physique peut être modifié indépendamment du niveau conceptuel. Cela signifie que tous les aspects matériels de la base de données n'apparaissent pas pour l'utilisateur, il s'agit simplement d'une structure transparente de représentation des informations.
- Indépendance logique : le niveau conceptuel doit pouvoir être modifié sans remettre en cause le niveau physique, c'est-à-dire que l'administrateur de la base doit pouvoir la faire évoluer sans que cela gêne les utilisateurs .



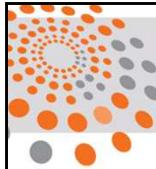


Rappel sur le modèle relationnel

Caractéristiques :

- Manipulabilité : des personnes ne connaissant pas la base de données doivent être capables de décrire leur requête sans faire référence à des éléments techniques de la base de données.
- Rapidité des accès : le système doit pouvoir fournir les réponses aux requêtes le plus rapidement possible, cela implique des algorithmes de recherche rapides.
- Administration centralisée : le SGBD doit permettre à l'administrateur de pouvoir manipuler les données, insérer des éléments, vérifier son intégrité de façon centralisée.



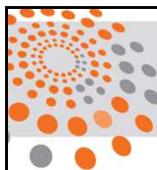


Rappel sur le modèle relationnel

Caractéristiques :

- Limitation de la redondance : le SGBD doit pouvoir éviter dans la mesure du possible des informations redondantes, afin d'éviter d'une part un gaspillage d'espace mémoire mais aussi des erreurs .
- Vérification de l'intégrité : les données doivent être cohérentes entre elles, de plus lorsque des éléments font référence à d'autres, ces derniers doivent être présents.



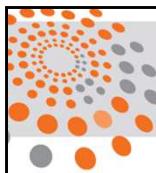


Rappel sur le modèle relationnel

Caractéristiques :

- Partageabilité des données : le SGBD doit permettre l'accès simultané à la base de données par plusieurs utilisateurs.
- Sécurité des données : le SGBD doit présenter des mécanismes permettant de gérer les droits d'accès aux données selon les utilisateurs.

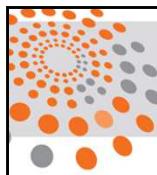




Rappel sur le modèle relationnel

Une requête est un ordre adressé à un SGBD. Cet ordre peut consister à extraire, à ajouter, à modifier, à administrer les données de la base. De façon générale, l'utilisateur, comme l'administrateur, dialogue avec le SGBD en lui soumettant des requêtes (des questions) et en récupérant en retour des résultats (les réponses).



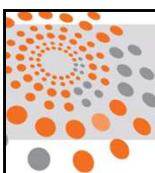


Introduction

Sommaire Module 1

- Rappel sur le modèle relationnel
- Les caractéristiques du langage SQL



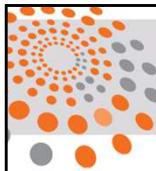


Langage SQL : les caractéristiques

Le langage Sql est devenu le standard en matière d'interface relationnelle, ceci probablement parce que ce langage est :

- issu de SEQUEL (interface de System-R), SQL a été développé chez IBM à San José ... !
- basé sur des mots clefs anglais explicites, il est relativement simple et facile à apprendre pour des utilisateurs non-informaticiens. Il illustre bien la tendance des langages formels à s'orienter vers un certain "langage naturel".
- normalisé





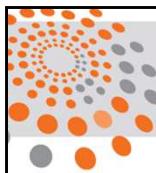
Langage SQL : les caractéristiques

Le standard Ansi a valeur nominative, en principe seulement aux Etats-Unis.

L'équivalent français est la norme Afnor. La norme internationale de Sql est la norme ISO (International Standards Organisation) numéro 9075 de 1987.

Les normes sont accompagnées de niveaux qui indiquent le degré d'évolution de SQL.



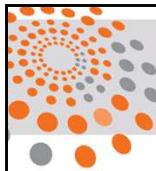


Langage SQL : les caractéristiques

L'ISO a défini les normes et les niveaux suivants :

- SQL86
- SQL89
- SQL92 (ou SQL2)
 - . Entry
 - . Transitional
 - . Intermediate
 - . Full
- SQL 1999 (ou SQL3)
- SQL 2003
- SQL 2008
- SQL 2011





Langage SQL : les caractéristiques

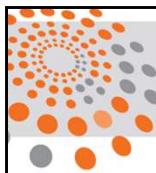
La norme définit deux langages SQL :

- un Langage de Manipulation de Données et de modules, (en anglais *SQLDML*), pour déclarer les procédures d'exploitation et les appels à utiliser dans les programmes.

On peut également rajouter une composante pour l'interrogation de la base : le Langage d'Interrogation de Données.

- un Langage de Définition de Données (en anglais *SQL-DDL*), à utiliser pour déclarer les structures logiques de données et leurs contraintes d'intégrité. On peut également rajouter une composante pour la gestion des accès aux données : le Langage de Contrôle de Données (en anglais *SQL-DCL*)



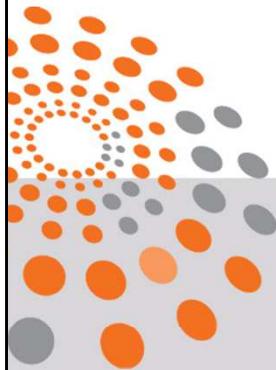


Langage SQL : les caractéristiques

Les instructions incontournables de SQL sont les suivantes :

Langage SQL			
LMD		LDD	
LID		LCD	
SELECT	INSERT	GRANT	CREATE
	UPDATE	REVOKE	ALTER
	DELETE		TRUNCATE
			DROP
			RENAME

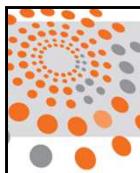




Module 2

Le Langage d'Interrogation de Données (LID)



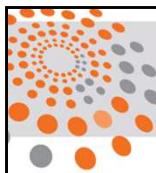


Le Langage d'Interrogation de données (LID)

Sommaire Module 2

- La Sélection de données
- Gestion des valeurs null
- Les restrictions ou conditions
- Les tris
- Les jointures



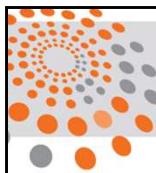


La commande SELECT

Le SELECT est la commande de base du SQL .

Elle permet d'extraire des données d'une base ou d'en calculer de nouvelles à partir de données existantes .





La commande SELECT

*SELECT [DISTINCT ou ALL] * ou liste de colonnes*

FROM nom de table ou de la vue

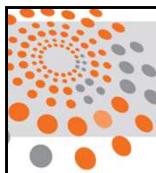
[WHERE prédictats]

[GROUP BY ordre des groupes]

[HAVING condition]

[ORDER BY liste de colonnes]





La commande SELECT

SELECT

Spécification des colonnes du résultat

FROM

Spécification des tables sur lesquelles porte le select

WHERE

Filtre portant sur les données (conditions à remplir pour faire afficher le résultat)

GROUP BY

Définition du sous ensemble

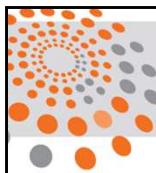
HAVING

conditions de regroupement des lignes

ORDER BY

Tri des données du résultat





La commande SELECT

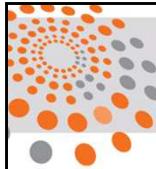
Requête:

```
SELECT CLI_NOM, CLI_PRENOM FROM T_CLIENT;
```

Résultat : CLI_NOM CLI_PRENOM

CLI_NOM	CLI_PRENOM
DUPONT	Alain
MARTIN	Marc
BOUVIER	Alain
DUBOIS	Paul
DREYFUS	Jean
FAURE	Alain
PAUL	Marcel
DUVAL	Arsène
PHILIPPE	André
CHABAUD	Daniel
BAILLY	Jean-François
...	



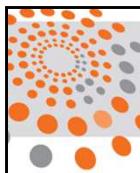


La commande SELECT

Il existe plusieurs opérateurs de sélection :

- Le caractère * récupère toutes les colonnes
- DISTINCT permet d'éliminer les doublons dans la réponse
- AS sert à donner un nom aux colonnes renvoyées par la requête
- L'opérateur || (double barre verticale) permet de concaténer des champs de type caractères (fonction concat ou le signe + dans certaines bases de données)
- On peut utiliser les opérateurs mathématiques de base pour combiner différentes colonnes (+, -, *, /)



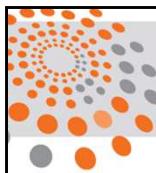


Le Langage d'Interrogation de Données (LID)

Sommaire Module 2

- La Sélection de données
- Gestion des valeurs null
- Les restrictions ou conditions
- Les tris
- Les jointures





La gestion des valeurs null

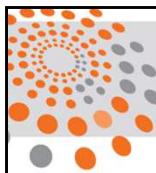
La gestion du null dans les bases de données :

COALESCE(expression testée, expression de remplacement)

Si l'argument *expression testée* n'est pas NULL, la fonction COALESCE() retournera la valeur de l'argument *expression testée*. Sinon elle retournera la valeur de l'argument *expression de remplacement*.

La fonction COLALESCE retourne une valeur numérique ou une chaîne de caractères.





La gestion des valeurs null

La gestion du null sous SQL Server

ISNULL(expression testée, expression de remplacement)

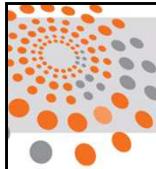
La gestion du null sous Oracle

NVL(expression testée, expression de remplacement)

La gestion du null sous MySQL

IFNULL(expression testée, expression de remplacement)





La gestion des valeurs null : exemples

Toutes bases> SELECT COALESCE(NULL,10) FROM...;

-> 10

MySQL> SELECT IFNULL(1,0) FROM...;

-> 1

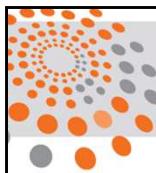
SQL Server>SELECT TITLE, ISNULL(PRICE, 0.00) FROM...;

-> 0,00

Oracle> SELECT NVL(SUPPLIER_CITY, 'N/A') FROM ...;

-> N/A

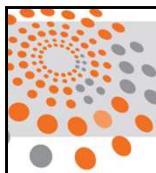




Atelier 2.1

1. Affichez tous les employés de la société.
2. Affichez toutes les catégories de produits.
3. Affichez le nom, le prénom, la date de naissance et la commission (à 0 si pas de commission) de tous les employées de la société.
4. Affichez la liste des fonctions des employés de la société.
5. Affichez la liste des pays de nos clients.
6. Affichez la liste des localités dans lesquelles il existe au moins un client.





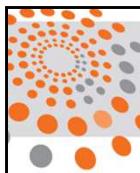
Atelier 2.2

1. Affichez les produits commercialisés et la valeur de stock par produit (prix unitaire * unités en stock).
2. Affichez le nom, le prénom, l'âge et l'ancienneté des employés.
3. Écrivez la requête qui permet d'afficher :

Employé a un gain annuel sur 12 mois

----- ----- ----- -----
Fuller gagne 120000 par an.



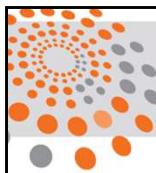


Le Langage d'Interrogation de Données (LID)

Sommaire Module 2

- La Sélection de données
- Gestion des valeurs null
- Les restrictions ou conditions
- Les tris
- Les jointures





La clause WHERE

Cette clause permet d'afficher des lignes vérifiant une condition.

Syntaxe :

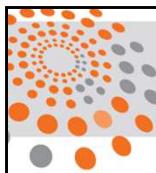
SELECT

FROM

WHERE *condition*

Ex: Select * from client where numclient=123;





La clause WHERE

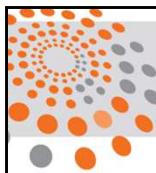
Dans la clause Where (clause conditionnelle), on peut utiliser tous les opérateurs logiques.

Where Expression opérateur Expression

L'opérateur peut être:

- =
- <, <=
- >, >=
- !=, <>, ^=



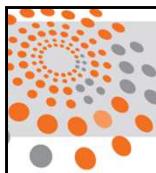


La clause WHERE

Il existe d'autres opérateurs :

- Opérateur IN
 - WHERE TIT_CODE IN ('Mme.', 'Melle.')
- Opérateur BETWEEN
 - WHERE CODE_POSTAL BETWEEN '91000' AND '91999'
- Opérateur LIKE
 - WHERE CLI_NOM LIKE 'B%'
- Comparaison logique
 - IS [NOT] {TRUE | FALSE} / IS [NOT] NULL
- Connecteurs logiques
 - OR | AND

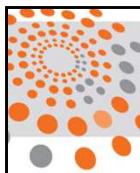




Atelier 2.3

1. Affichez le nom de la société et le pays des clients qui habitent à Toulouse.
2. Affichez le nom, le prénom et la fonction des employés dirigés par l'employé numéro 2.
3. Affichez le nom, le prénom et la fonction des employés qui ne sont pas des représentants.
4. Affichez le nom, le prénom et la fonction des employés qui ont un salaire inférieur à 3500.
5. Affichez le nom, la ville et le pays des clients qui n'ont pas de fax.
6. Affichez le nom, le prénom et la fonction des employés qui n'ont pas de supérieur.



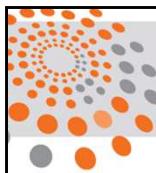


Le Langage d'Interrogation des Données (LID)

Sommaire Module 2

- La Sélection de données
- Gestion des valeurs null
- Les restrictions ou conditions
- Les tris
- Les jointures





La clause ORDER BY

Cette clause permet de définir le tri des colonnes.

ASC spécifie l'ordre ascendant et DESC l'ordre descendant du tri. ASC ou DESC peut être omis, dans ce cas c'est l'ordre ascendant qui est utilisé par défaut.

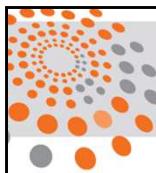
SELECT

FROM

WHERE

ORDER BY;





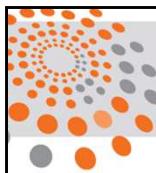
La clause ORDER BY

```
SELECT CLI_NOM, CLI_PRENOM  
FROM T_CLIENT  
ORDER BY CLI_NOM, CLI_PRENOM;
```

ou

```
SELECT CLI_NOM, CLI_PRENOM  
FROM T_CLIENT  
ORDER BY 1, 2;
```

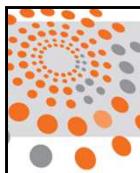




Atelier 2.4

1. Trier les employés par nom en ordre décroissant (afficher le nom et le prénom).
2. Trier les clients par pays (afficher le nom des sociétés et la ville).
3. Trier les clients par pays et par ville (afficher le nom des sociétés, le pays et la ville).
4. Trier les employés par commission (afficher le nom et la commission).



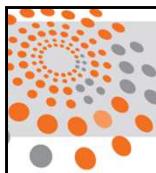


Le Langage d'Interrogation des Données (LID)

Sommaire Module 2

- La Sélection de données
- Gestion des valeurs null
- Les restrictions ou conditions
- Les tris
- Les jointures





Jointures

Jointures entre deux tables.

Soient deux tables *table1* et *table2*. *table1* a les colonnes *col1* et *col2* et *table2* les colonnes *cola*, *colb*.

Supposons que le contenu des tables soit le suivant :

table1	col1	col2
x	3	
y	4	

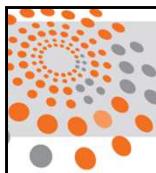
table2	cola	colb
a	7	
b	4	

Soit la commande :

```
SELECT col1, cola FROM table1, table2  
WHERE table1.col2=table2.colb;
```

Cette requête extrait des données de deux tables : *table1* et *table2* avec une condition entre deux colonnes de tables différentes.





Jointures

Comment fonctionne-t-elle ?

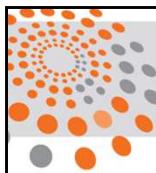
Une nouvelle table est construite avec pour colonnes l'ensemble des colonnes des deux tables et pour lignes le produit cartésien des deux tables :

col1	col2	cola	colb
x	3	a	7
x	3	b	4
y	4	a	7
y	4	b	4

La condition *WHERE col2=colb* est appliquée à cette nouvelle table. On obtient donc la nouvelle table suivante :

col1	col2	cola	colb
y	4	b	4



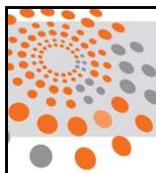


Jointures

Il y a ensuite affichage des colonnes demandées (select) :

col1	cola
y	b



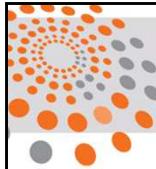


Jointures

Syntaxe d'une requête multi-tables :

```
SELECT colonne1, colonne2, ...
FROM table1, table2, ..., tablep
WHERE jointure1
      AND jointure2
      AND ...
ORDER BY ...;
```





Jointures

- Sans condition (produit cartésien) :

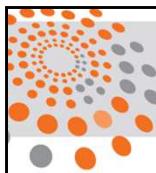
```
SELECT CLI_NOM, TEL_NUMERO  
FROM T_CLIENT, T_TELEPHONE;
```

- Avec condition :

```
SELECT CLI_NOM, TEL_NUMERO  
FROM T_CLIENT, T_TELEPHONE  
WHERE T_CLIENT.CLI_ID = T_TELEPHONE.CLI_ID;
```

```
SELECT CLI_NOM, TEL_NUMERO  
FROM T_CLIENT C, T_TELEPHONE T  
WHERE C.CLI_ID = T.CLI_ID AND TYP_CODE = 'FAX';
```





Jointures

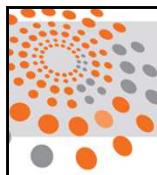
Jointure naturelle

```
SELECT ...
FROM <table gauche>
      NATURAL JOIN <table droite>
;
```

```
SELECT CLI_NOM, TEL_NUMERO
FROM T_CLIENT NATURAL JOIN T_TELEPHONE;
```

La jointure se fait sur la colonne qui porte le même nom dans les deux tables.





Jointures

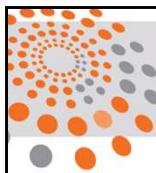
Jointure naturelle

```
SELECT ...
FROM <table gauche>
JOIN <table droite>
USING <noms de colonnes>
;
```

```
SELECT CLI_NOM, TEL_NUMERO
FROM T_CLIENT JOIN T_TELEPHONE USING CLI_ID;
```

La jointure se fait sur la colonne qui porte le même nom dans les deux tables.





Jointures

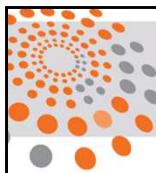
Jointure
interne

```
SELECT ...
FROM <table gauche>
[INNER] JOIN <table droite>
ON <condition de jointure>
;
```

```
SELECT CLI_NOM, TEL_NUMERO
FROM T_CLIENT INNER JOIN T_TELEPHONE
ON T_CLIENT.CLI_ID = T_TELEPHONE.CLI_ID;
```

Ici on spécifie le nom des colonnes sur lesquelles se fait la jointure.





Jointures

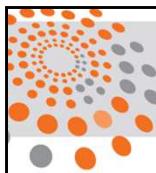
Jointure externe

```
SELECT ...
FROM <table gauche>
LEFT | RIGHT | FULL [OUTER] JOIN <table droite>
ON condition de jointure
```

```
SELECT CLI_NOM, TEL_NUMERO
FROM T_CLIENT C LEFT OUTER JOIN T_TELEPHONE T
ON C.CLI_ID = T.CLI_ID AND TYP_CODE IS NULL ;
```

Les jointures externes rapatrient les informations disponibles, même si des lignes de la table ne sont pas renseignées entre les différentes tables jointes





Jointures

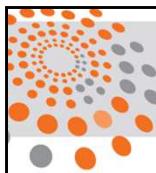
SELECT colonnes

FROM TGauche LEFT OUTER JOIN TDroite

ON condition de jointure;

On recherche toutes les valeurs satisfaisant la condition de jointure précisée dans le prédictat, puis on rajoute toutes les lignes de la table TGauche qui n'ont pas été prises en compte au titre de la satisfaction du critère.





Jointures

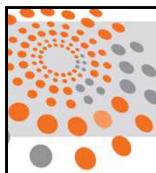
SELECT colonnes

FROM TGauche RIGHT OUTER JOIN TDroite

ON condition de jointure

On recherche toutes les valeurs satisfaisant la condition de jointure précisée dans prédictat, puis on rajoute toutes les lignes de la table TDroite qui n'ont pas été prises en compte au titre de la satisfaction du critère.





Jointures

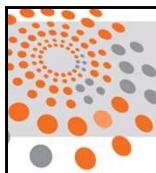
SELECT colonnes

FROM TGauche FULL OUTER JOIN TDroite

ON condition de jointure

On recherche toutes les valeurs satisfaisant la condition de jointure précisée dans prédicat, puis on rajoute toutes les lignes de la table TGauche et TDroite qui n'ont pas été prises en compte au titre de la satisfaction du critère.





Jointures

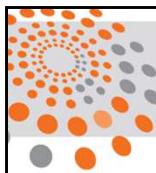
Jointure croisée

```
SELECT ...
FROM <table gauche>
      CROSS JOIN <table droite>
```

```
SELECT CHB_ID, TRF_DATE_DEBUT, 0 AS TRF_CHB_PRIX
FROM T_TARIF CROSS JOIN T_CHAMBRE
ORDER BY CHB_ID, TRF_DATE_DEBUT;
```

On fait sciemment le produit cartésien .

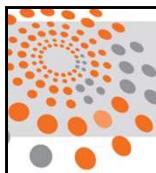




Atelier 2.5

1. Affichez le nom, le prénom, la fonction et le salaire des employés qui ont un salaire compris entre 2500 et 3500.
2. Affichez le nom du produit, le nom du fournisseur, le nom de la catégorie et les quantités de produits qui ne sont pas d'une des catégories 1,3,5 et 7.
3. Affichez le nom du produit, le nom du fournisseur, le nom de la catégorie et les quantités des produits qui ont un numéro de fournisseur entre 1 et 3 ou un code catégorie entre 1 et 3, et pour lesquelles les quantités sont données en boîte(s) ou en carton(s).

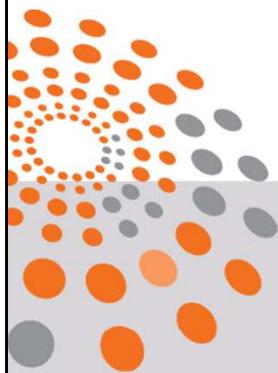




Atelier 2.5

4. Écrivez la requête qui permet d'afficher le nom des employés qui ont effectué au moins une vente pour un client parisien.
5. Affichez le nom des produits et le nom des fournisseurs pour les produits des catégories 1,4 et 7.
6. Affichez le nom des employés ainsi que le nom de leur supérieur hiérarchique.

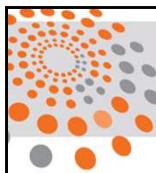




Module 3

Utilisation de fonctions





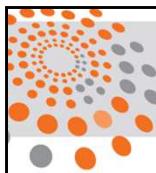
Fonction de transtypage

Transtypage :

Il permet de changer le type de données d'une colonne afin d'effectuer une comparaison de données de type hétérogène.

```
SELECT CHB_ID, CHB_NUMERO, CHB_POSTE_TEL  
FROM T_CHAMBRE  
WHERE  
CAST(CHB_POSTE_TEL AS INTEGER) / 10 > CHB_NUMERO ;
```





Fonctions de chaînes de caractères

Mise en majuscule / Minuscule :

Les opérateurs LOWER et UPPER permettent de mettre en majuscule ou en minuscule des chaînes de caractères dans les requêtes.

```
SELECT upper(CLIENT_PRENOM), lower(CLIENT_NOM)  
FROM T_CLIENT;
```

CLI_NOM CLI_PRENOM

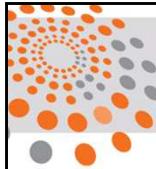
ALAIN dupont

MARC martin

ALAIN bouvier

.....





Fonctions de chaînes de caractères

Extraire une sous chaîne :

La fonction SUBSTRING (SUBSTR sous Oracle et Postgre) permet d'extraire une sous chaîne d'une chaîne de caractère.

Exemple SQL Server :

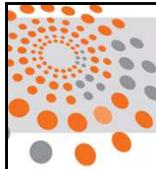
```
SELECT CLI_NOM, CLI_PRENOM, SUBSTRING(CLI_PRENOM,1,1) +  
SUBSTRING (CLI_NOM,1,1) AS INITIALES FROM T_CLIENT;
```

CLI_NOM CLI_PRENOM INITIALES

DUPONT Alain AD

MARTIN Marc MM





Fonctions date système

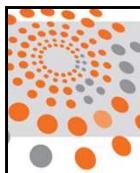
Heure et date courante :

Exemple Postgre, MySql, Oracle :

```
SELECT NO_COMMANDE  
FROM commandes WHERE  
DATE_ENVOI BETWEEN CURRENT_DATE  
and CURRENT_DATE - 14;
```

Oracle	SYSDATE CURRENT_DATE
Sybase	GETDATE()
SQL Server	GETDATE()
Access	DATE()
MySQL Postgre	CURRENT_DATE



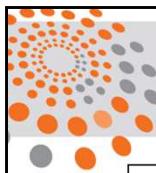


Fonctions d'agrégation (ou de regroupement)

Fonctions statistiques :

```
SELECT AVG(TRF_CHB_PRIX) as MOYENNE,  
MAX(TRF_CHB_PRIX) as MAXI,  
MIN(TRF_CHB_PRIX) as MINI,  
SUM(TRF_CHB_PRIX) as TOTAL,  
COUNT(TRF_CHB_PRIX) as NOMBRE  
FROM TJ_TRF_CHB ;
```

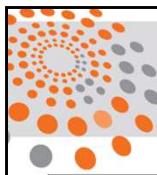




Fonctions mathématiques

ABS(expression)	valeur absolue
MOD(expression)	modulo
SIGN(expression)	signe
SQRT(expression)	racine carrée
CEIL(expression) ; Sql Server : ceiling(expression)	plus grand entier
FLOOR(expression)	plus petit entier
ROUND(expression, nb_décimales)	arrondi
TRUNC(expression, nb_décimales) Sql Server : ROUND(expression, nb_décimales,1)	tronqué
EXP(expression)	exponentielle
LN(expression)	logarithme népérien
LOG(expression)	logarithme décimal
POWER(expression,valeur)	puissance
COS(expression) ; SIN((expression))	cosinus ; sinus
TAN(expression)	tangente
PI() ; Oracle : ASIN(1)*2	constante Pi

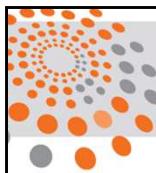




Fonctions de chaînes de caractères

CONCAT(expression1,expression2,...)	concaténation : utiliser de préférence chez Oracle (car deux arguments maximum avec la fonction)
INITCAP(expression) (Oracle, Postgre)	initiales en lettres capitales
LPAD(expression,nb_caractères,caractère) (Mysql ,Oracle, Postgre)	complément ou troncature à n position à gauche
RPAD(expression,nb_caractères,caractère) (Mysql ,Oracle, Postgre)	complément ou troncature à n position à droite
LTRIM(expression) RTRIM(expression)	suppression des espaces en tête/queue d'une chaîne
REPLACE (expression, valeur à remplacer, valeur de remplacement)	remplacement
INSTR(expression,'texte cherché') (Mysql ,Oracle)	position d'une chaîne dans une sous chaîne
PATINDEX('%texte cherché%', expression) (Sql Server)	position d'une chaîne dans une sous chaîne
POSITION('texte cherché' in expression) (Postgre)	position d'une chaîne dans une sous chaîne

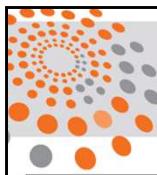




Fonctions de chaînes de caractères

LENGTH(expression) (Mysql ,Oracle, Postgre)	longueur de la chaîne
LEN (expression) (Sql Server)	longueur de la chaîne
ASCII(expression)	code ASCII d'un caractère
CHR(valeur) (Oracle et Postgre)	caractère dont le code ASCII est donné
CHAR(valeur) (Mysql et Sql Server)	caractère dont le code ASCII est donné
REVERSE(expression)	Inverse l'ordre des caractères d'une chaîne

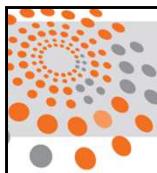




Fonctions de dates

Expression + INTERVAL '1' day (Mysql ,Oracle, Postgre)	ajoute des jours, des mois, des années à une date
DATEADD (day,1,expression) (Sql Server)	ajoute des jours, des mois, des années à une date
ADD_MONTHS(expression, nb_mois) (Oracle)	ajoute des mois à une date
NEXT_DAY (expression,'jour') (Oracle)	date du prochain jour d'un nom donné
LAST_DAY(expression) (Oracle, Mysql)	renvoie le n° du dernier jour d'un mois d'une date
MONTHS_BETWEEN(date1,date2) (Oracle)	nombre de mois entre deux dates
DATEDIFF (Sql Server et Mysql)	différence entre deux dates
DATEPART(day,expression) (Sql Server)	Renvoie la valeur du jour, mois ou année
EXTRACT(day from expression) (Mysql ,Oracle, Postgre)	Renvoie la valeur du jour, mois ou année

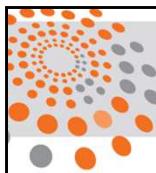




Fonctions de transtypage

TO_CHAR(expression,'format') (Oracle et Postgre)	date sous forme littérale
TO_DATE(expression,'format') (Oracle et Postgre)	Convertit une chaîne de caractère en date
CONVERT(type, expression) (Sql Server)	Convertit une chaîne de caractère au format souhaité
CAST (expression as format)	Convertit une chaîne de caractère au format souhaité

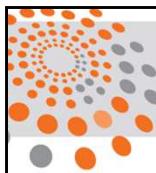




Atelier 3.1

1. Affichez la somme des salaires et des commissions des employés (cumuler les 2 en une seule colonne).
2. Affichez la moyenne des salaires et la moyenne des commissions des employés.
3. Affichez le salaire maximum et la plus petite commission des employés.
4. Affichez le nombre distinct de fonction.

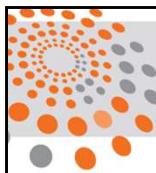




La clause GROUP BY

La clause GROUP BY est nécessaire dès que l'on utilise des fonctions de calculs statistiques avec des données brutes. Cette clause groupe les lignes en se basant sur la valeur de colonnes spécifiées et renvoie une seule ligne par groupe.





La clause GROUP BY

Sans group by :

```
SELECT COUNT(CHB_ID) AS NOMBRE, CHB_ETAGE FROM T_CHAMBRE;
```

NOMBRE CHB_ETAGE

1 RDC

1 RDC

1 RDC

1 RDC

1 1er

1 1er

1 1er

.....





La clause GROUP BY

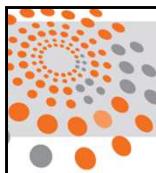
Avec Group By :

```
SELECT COUNT(CHB_ID) AS NOMBRE, CHB_ETAGE FROM T_CHAMBRE  
GROUP BY CHB_ETAGE;
```

NOMBRE CHB_ETAGE

8 1er
8 2e
4 RDC





La clause HAVING

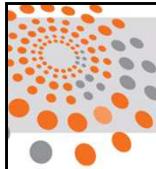
La clause HAVING remplace le WHERE sur les opérations résultant des regroupements

```
SELECT SUM(CHB_COUCHAGE) AS NOMBRE,  
CHB_ETAGE  
FROM T_CHAMBRE  
GROUP BY CHB_ETAGE  
HAVING SUM(CHB_COUCHAGE) >= 20;
```

NOMBRE CHB_ETAGE

23 1er
22 2e

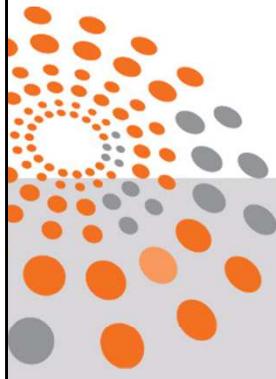




Atelier 3.2

1. Écrivez la requête qui permet d'afficher la masse salariale des employés par fonction.
2. Affichez le montant de chaque commande, en ne conservant que les commandes qui comportent plus de 5 références de produit.
3. Afficher la valeur des produits en stock et la valeur des produits commandés par fournisseur, pour les fournisseurs qui ont un numéro compris entre 3 et 6.

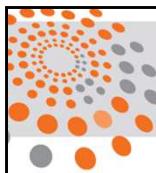




Module 4

Opérateurs ensemblistes





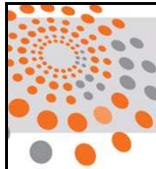
Les opérateurs ensemblistes

Les opérations ensemblistes en SQL sont celles définies dans l'algèbre relationnelle. Elles sont réalisées grâce aux opérateurs :

- UNION
- INTERSECT (n'est pas implémenté dans tous les SGBD)
- EXCEPT ou MINUS (n'est pas implémenté dans tous les SGBD)

Ces opérateurs s'utilisent entre deux clauses *SELECT*.





Les opérateurs assemblistes

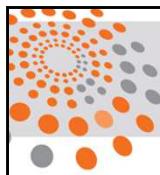
L'opérateur UNION :

Cet opérateur permet d'effectuer une UNION des lignes sélectionnées par deux clauses SELECT.

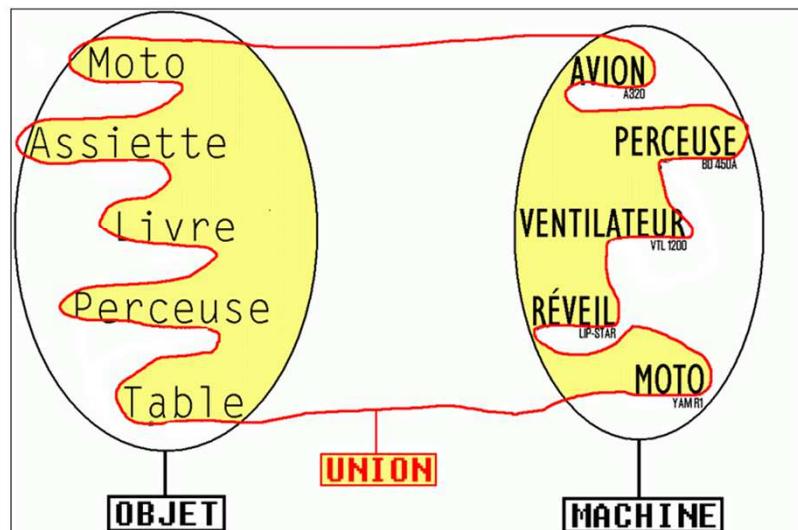
```
SELECT ---- FROM ---- WHERE ----- UNION  
SELECT ---- FROM ---- WHERE ----- ;
```

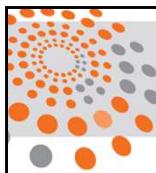
Par défaut les doublons sont automatiquement éliminés. Pour conserver les doublons, il est possible d'utiliser une clause UNION ALL.





Opérateur Union





Les opérateurs ensemblistes

L'opérateur INTERSECT :

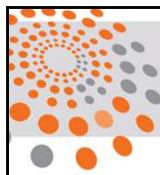
Cet opérateur permet d'effectuer une INTERSECTION des enregistrements sélectionnés par deux clauses SELECT.

```
SELECT ---- FROM ---- WHERE ----- INTERSECT  
SELECT ---- FROM ---- WHERE ----- ;
```

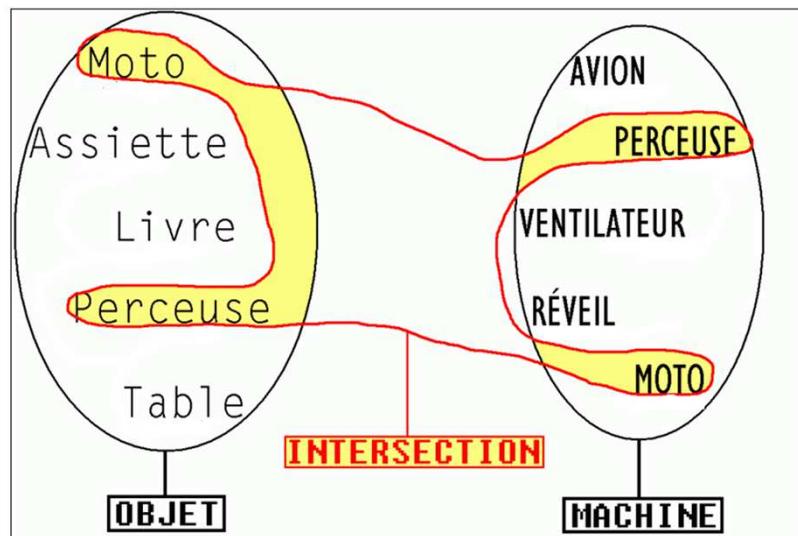
L'opérateur INTERSECT n'étant pas implémenté dans tous les SGBD, il est possible de le remplacer par des commandes usuelles (sous-interrogation):

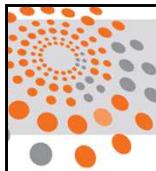
```
SELECT a,b FROM table1 WHERE EXISTS ( SELECT c,d FROM  
table2 WHERE a=c AND b=d );
```





Opérateur Intersect





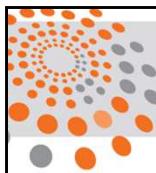
Les opérateurs ensemblistes

L'opérateur MINUS (ORACLE) ou EXCEPT :

Cet opérateur permet d'effectuer une DIFFERENCE entre les enregistrements sélectionnés par deux clauses SELECT, c'est-à-dire sélectionner les enregistrements de la première table n'appartenant pas à la seconde.

```
SELECT a,b FROM table1 WHERE ----- EXCEPT  
SELECT c,d FROM table2 WHERE -----;
```



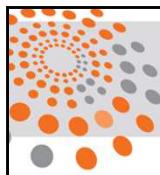


Les opérateurs ensemblistes

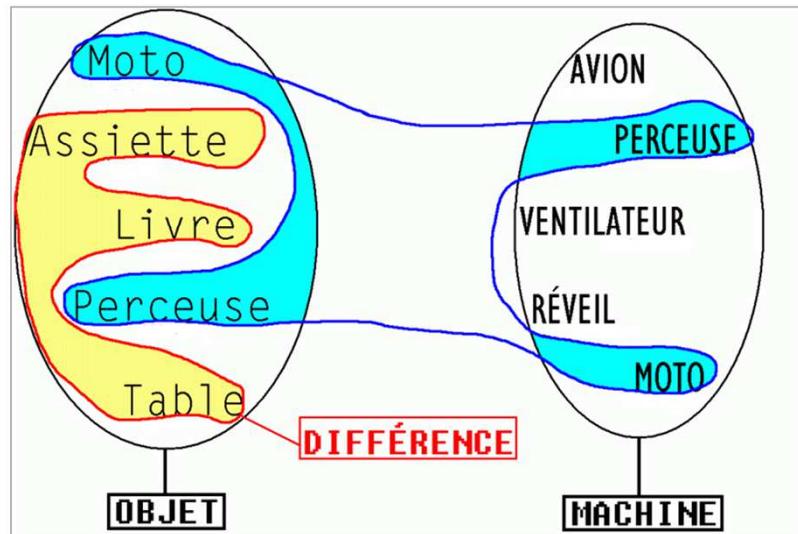
L'opérateur EXCEPT (ou MINUS) n'étant pas implémenté dans tous les SGBD, il est possible de le remplacer par des commandes usuelles (en l'espèce une sous-interrogation) :

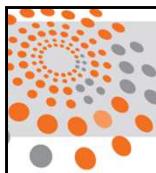
```
SELECT a,b FROM table1  
WHERE NOT EXISTS (SELECT c,d FROM table2  
                   WHERE a=c AND b=d );
```





Opérateur Minus (ou Except)

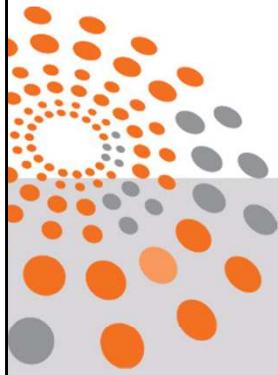




Atelier 4

1. Affichez les sociétés, adresse et villes de résidence pour tous les tiers de l'entreprise.
2. Affichez toutes les commandes qui comportent en même temps des produits de catégorie 1 du fournisseur 1 et des produits de catégorie 2 du fournisseur 2.
3. Affichez la liste des produits que les clients parisiens ne commandent pas.

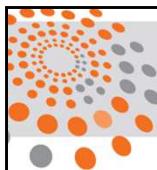




Module 5

Sous-interrogations



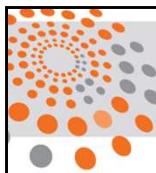


Sous-interrogations

Sommaire Module 5

- Dans la clause WHERE
- Dans la clause FROM
- Sous-interrogations synchronisées





Dans la clause Where

Une caractéristique puissante de SQL est la possibilité qu'un prédictat employé dans une clause WHERE (expression à droite d'un opérateur de comparaison) comporte un SELECT emboîté.

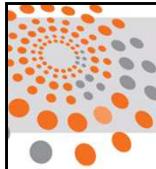
Sous-interrogation à une ligne et une colonne

Dans ce cas, le SELECT imbriqué équivaut à une valeur.

WHERE exp op (SELECT ...)

où op est un des opérateurs : =, !=, <>, <, >, <=, >= [not] in





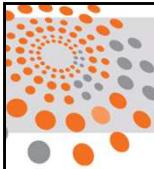
Dans la clause Where

Exemple :

Liste des employés travaillant dans le même département que MERCIER :

```
SELECT NOME FROM EMP  
WHERE DEPT = (SELECT DEPT FROM EMP  
               WHERE NOME = 'MERCIER');
```





Dans la clause Where

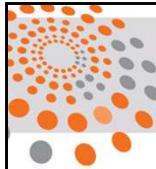
Sous-interrogation ramenant plusieurs lignes

Une sous-interrogation peut ramener plusieurs lignes à condition que l'opérateur de comparaison admette à sa droite un ensemble de valeurs.

Les opérateurs permettant de comparer une valeur à un ensemble de valeurs sont :

- l'opérateur IN
- les opérateurs obtenus en ajoutant ANY ou ALL à la suite des opérateurs de comparaison classique =, <>, <, >, <=, >=
 - ANY : la comparaison sera vraie si elle est vraie pour au moins un élément de l'ensemble (elle est donc fausse si l'ensemble est vide). On remplacera any par la fonction MIN dans la sous-requête.
 - ALL : la comparaison sera vraie si elle est vraie pour tous les éléments de l'ensemble (elle est vraie si l'ensemble est vide).
On remplacera all par la fonction MAX dans la sous-requête.





Dans la clause Where

- WHERE exp op ANY (SELECT ...)
 - WHERE exp op ALL (SELECT ...)
- où op est un des opérateurs =, !=, <, >, <=, >=
- WHERE exp IN (SELECT ...)
 - WHERE exp NOT IN (SELECT ...)

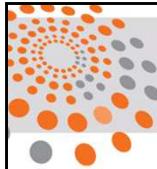
Liste des employés gagnant plus que tous les employés du département 30 :

```
SELECT NOME, SAL FROM EMP  
WHERE SAL > ALL (SELECT SAL FROM EMP  
                  WHERE DEPT=30);
```

Ou :

```
SELECT NOME, SAL FROM EMP  
WHERE SAL > (SELECT MAX(SAL) FROM EMP  
                  WHERE DEPT=30);
```



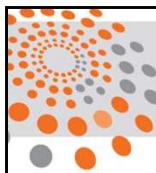


Sous-interrogations

Sommaire Module 5

- Dans la clause WHERE
- Dans la clause FROM
- Sous-interrogations synchronisées



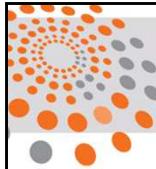


Dans la clause From

Syntaxe :

```
Select A.x , A.y, B.z  
from table A , ( select x, z from table ) B  
Where A.x = B.x;  
Ou  
Select A.x , A.y, B.z  
from table A  
Join ( select x, z from table ) B on A.x = B.x;
```





Dans la clause From

Exemple : part du salaire de chaque employé par rapport à la masse salariale.

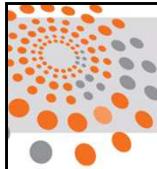
```
Select nom, salaire, round(salaire/masse*100)  
from employes, (select sum(salaire) masse from employes) b;
```

Ou avec la fonction over() :

```
Select nom, salaire, salaire/sum(salaire) over()*100 from  
employes;
```

Over() ne fonctionne pas sous MYSQL.



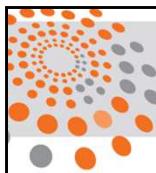


Sous-interrogations

Sommaire Module 5

- Dans la clause WHERE
- Dans la clause FROM
- Sous-interrogations synchronisées

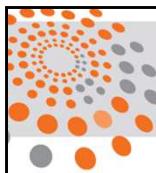




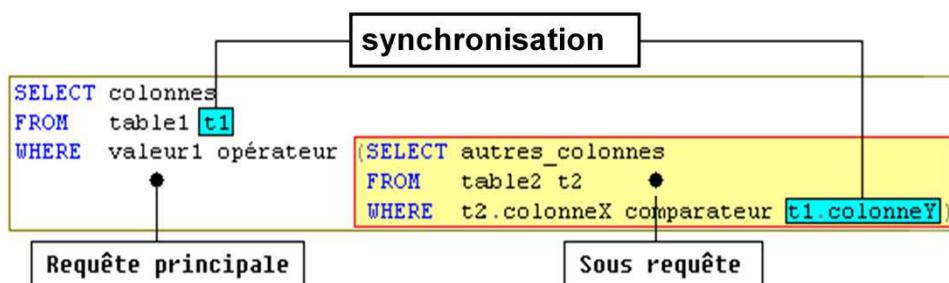
Sous-requête synchronisée

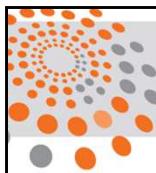
- Une sous requête synchronisée est une sous requête qui s'exécute pour chaque ligne de la requête principale et non une fois pour toute.
- Pour arriver à ce résultat, il suffit de faire varier une condition en rappelant dans la sous requête la valeur de la ou des colonnes de la requête principale qui doit servir de condition.





Sous-requête synchronisée

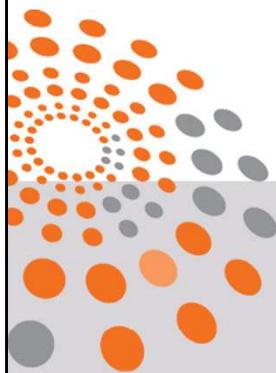




Atelier 5

1. Affichez tous les produits pour lesquels la quantité en stock est inférieure à la moyenne des quantités en stock.
2. Affichez toutes les commandes pour lesquelles les frais de ports dépassent la moyenne des frais de ports pour ce client.
3. Affichez les produits pour lesquels la quantité en stock est supérieure à la quantité en stock de chaque produit de catégorie 3.

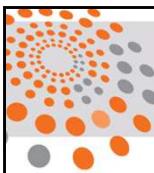




Module 6

Le Langage de Manipulation de Données (LMD)





LMD

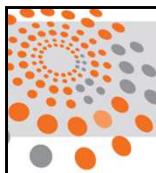
Le langage de manipulation de données (LMD) est le langage permettant de modifier les informations contenues dans la base.

Il existe trois commandes SQL permettant d'effectuer les trois types de modification des données :

- INSERT = ajout de lignes
- UPDATE = mise à jour de lignes
- DELETE = suppression de lignes

Ces trois commandes peuvent être effectuées dans le cadre d'une transaction. En mode transactionnel, on peut confirmer les modifications (COMMIT) ou les annuler (ROLLBACK)



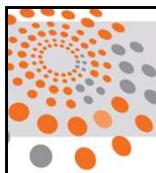


Insertion

```
INSERT INTO table (col1,..., coln )
VALUES (val1,...,valn )
ou
INSERT INTO table (col1,..., coln )
SELECT ...
```

table est le nom de la table sur laquelle porte l'insertion.
col1,..., coln est la liste des noms des colonnes pour lesquelles on donne une valeur. Cette liste est optionnelle. Si elle est omise, le SGBD prendra par défaut l'ensemble des colonnes de la table dans l'ordre où elles ont été données lors de la création de la table. Si une liste de colonnes est spécifiée, les colonnes ne figurant pas dans la liste auront la valeur NULL.





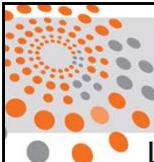
Insertion

```
INSERT INTO dept  
VALUES (10, 'FINANCES', 'PARIS');
```

```
INSERT INTO dept (lieu, nomd, dept)  
VALUES ('GRENOBLE', 'RECHERCHE', 20);
```

```
INSERT INTO PARTICIPATION (MATR, CODEP)  
SELECT MATR, 10 FROM EMP  
WHERE NOME = 'MARTIN';
```





Modification

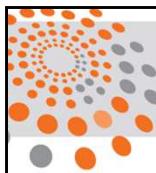
La commande UPDATE permet de modifier les valeurs d'un ou plusieurs champs, dans une ou plusieurs lignes existantes d'une table.

```
UPDATE table  
SET col1 = exp1, col2 = exp2, ...  
WHERE prédicat;  
ou  
UPDATE table  
SET (col1, col2,...) = (SELECT ...)  
WHERE prédicat;
```

table est le nom de la table mise à jour ; col1, col2, ... sont les noms des colonnes qui seront modifiées ; exp1, exp2,... sont des expressions. Elles peuvent aussi être un ordre SELECT renvoyant les valeurs attribuées aux colonnes (deuxième variante de la syntaxe).

Les valeurs de col1, col2... sont mises à jour dans toutes les lignes satisfaisant le prédictat. La clause WHERE est facultative. Si elle est absente, toutes les lignes sont mises à jour.





Modification

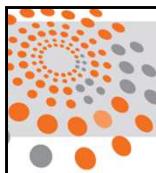
Faire passer MARTIN dans le département 10 :

```
UPDATE EMP  
SET DEPT = 10  
WHERE NOME = 'MARTIN';
```

Donner à CLEMENT un salaire 10 % au dessus de la moyenne des salaires des secrétaires :

```
UPDATE EMP  
SET SAL =      (SELECT AVG(SAL) * 1.10  
                 FROM EMP  
                 WHERE POSTE = 'SECRETAIRE')  
WHERE NOME = 'CLEMENT';
```





Suppression

L'ordre DELETE permet de supprimer des lignes d'une table.

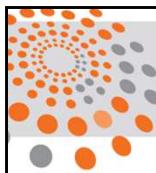
`DELETE FROM table`

`WHERE prédicat;`

La clause WHERE indique quelles lignes doivent être supprimées.

ATTENTION : cette clause est facultative ; si elle n'est pas précisée, TOUTES LES LIGNES DE LA TABLE SONT SUPPRIMEES (heureusement qu'il existe ROLLBACK!).



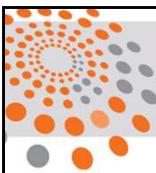


Suppression

```
DELETE FROM dept  
WHERE dept = 10;
```

DELETE FROM dept; <= j'efface toutes les lignes

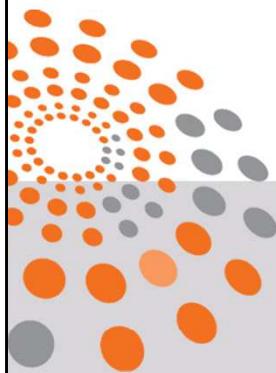




Atelier 6

1. Insérez une nouvelle catégorie de produits nommée «fruits et légumes », en respectant les contraintes.
2. Créez un nouveau fournisseur « Grandma » (no_fournisseur = 30) avec les mêmes coordonnées que le fournisseur « Grandma Kelly's Homestead ».
3. Attribuer les produits de « Grandma Kelly's Homestead » au nouveau fournisseur précédemment créé.
4. Supprimez l'ancien fournisseur « Grandma Kelly's Homestead » .

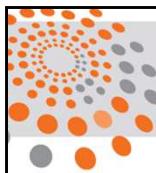




Module 7

Notions sur le Langage de Définition de données (LDD)





Types de données

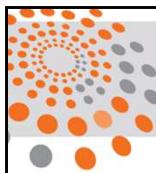
CHAR, CHARACTER, VARCHAR, NCHAR, NVARCHAR

DECIMAL, NUMERIC, FLOAT, REAL, DOUBLE, SMALLINT, INTEGER

BIT, NBIT, BLOB, IMAGE

TIMESTAMP, DATE, TIME, INTERVAL





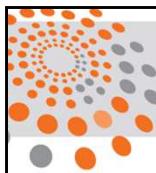
Création de table

```
CREATE TABLE table  
(colonne1 type1 contraintes,  
colonne2 type2 contraintes,  
.....  
.....  
, contraintes de table)  
;
```

table est le nom que l'on donne à la table ;

colonne1, colonne2,... sont les noms des colonnes ; type1, type2,... sont les types des données qui seront contenues dans les colonnes.

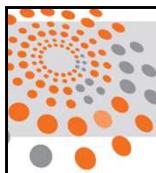




Contraintes

- Contraintes de colonne :
[NOT] NULL | UNIQUE | PRIMARY KEY |
CHECK (prédicat_de_colonne) |
FOREIGN KEY [colonne] REFERENCES table (colonne)
- Contraintes de table :
CONSTRAINT nom_contrainte UNIQUE |
PRIMARY KEY (liste_colonne) |
CHECK (prédicat_de_table) |
FOREIGN KEY liste_colonne REFERENCES nom_table
(liste_colonne)





Contraintes

Une colonne peut donc recevoir les contraintes suivantes :

NULL / NOT NULL : précise si la valeur est obligatoire.

DEFAULT : valeur par défaut qui est placée dans la colonne lors des insertions et de certaines opérations particulières, lorsque l'on n'a pas donné de valeur explicite à la colonne.

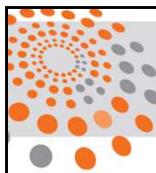
PRIMARY KEY : précise si la colonne est la clé primaire de la table.

UNIQUE : les valeurs de la colonne doivent être uniques (pas de doublon)

CHECK : permet de préciser un prédictat qui acceptera la valeur s'il est vérifié

FOREIGN KEY : permet, pour les valeurs de la colonne, de faire référence à des valeurs existantes dans une colonne d'une autre table. Ce mécanisme s'appelle intégrité référentielle.



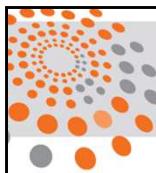


Création de table

Exemple:

```
CREATE TABLE T_CLIENT
(
    CLI_ID INTEGER NOT NULL PRIMARY KEY,
    CLI_NOM CHAR(32) NOT NULL,
    CLI_PRENOM VARCHAR(32)
);
```



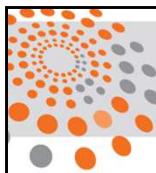


Création de table

Exemple:

```
CREATE TABLE T_VOITURE
(
    VTR_ID INTEGER NOT NULL PRIMARY KEY,
    VTR_MARQUE CHAR(32) NOT NULL,
    VTR_MODELE VARCHAR(16),
    VTR_IMMATRICULATION CHAR(10) NOT NULL UNIQUE,
    VTR_COULEUR CHAR(5) CHECK (VALUE IN ('BLANC', 'NOIR',
    'ROUGE', 'VERT', 'BLEU'))
);
```





Modifier une table

`ALTER TABLE nom_table`

Ajout d'une colonne - ADD :

`ALTER TABLE table ADD col1 type1...;`

Modification d'une colonne - Oracle et MySQL :

`ALTER TABLE table MODIFY col1 type1, ...;`

Sql Server et Postgre :

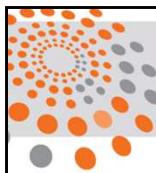
`ALTER TABLE table ALTER COLUMN col1 type1, ...;`

Suppression d'une colonne - DROP COLUMN

`ALTER TABLE table DROP COLUMN col;`

La colonne supprimée ne doit pas être référencée par une clé étrangère ou être utilisée par un index.





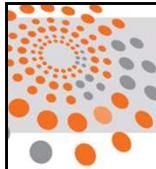
Modifier une table

```
ALTER TABLE T_CLIENT  
ADD CLI_PRENOM VARCHAR(25);
```

```
ALTER TABLE T_CLIENT  
ADD CLI_DATE_NAISSANCE DATE;
```

```
ALTER TABLE T_CLIENT ADD CONSTRAINT  
CHK_DATE_NAISSANCE CHECK (CLI_DATE_NAISSANCE BETWEEN  
'1880-01-01' AND '2020-01-01');
```





Modifier une table

Ajouter, supprimer ou renommer une contrainte :

Des contraintes d'intégrité peuvent être ajoutées ou supprimées par la commande ALTER TABLE. On ne peut ajouter que des contraintes de table.

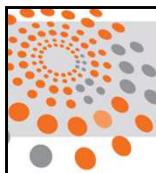
```
ALTER TABLE EMP DROP CONSTRAINT NOM_UNIQUE;
```

```
ALTER TABLE EMP ADD CONSTRAINT SAL_MIN CHECK(SAL +  
NVL(COMM,0) > 1000);
```

```
ALTER TABLE EMP MODIFY CONSTRAINT SAL_MIN DISABLE;
```

```
ALTER TABLE EMP RENAME CONSTRAINT NOM1 TO NOM2;
```



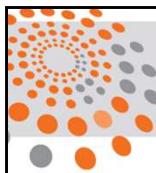


Supprimer une table

```
DROP TABLE nom_objet;
```

```
DROP TABLE TMP_IMP_DATE;
```





Les vues

Une vue est une vision des données d'une ou plusieurs tables de la base.

La définition d'une vue est donnée par un SELECT qui indique les données de la base qui seront vues.

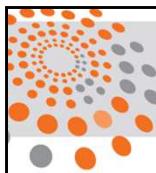
La commande CREATE VIEW permet de créer une vue en spécifiant le SELECT constituant la définition de la vue :

```
CREATE VIEW vue (col1, col2...) AS SELECT ...;
```

Exemple:

```
CREATE VIEW EMP10 AS SELECT * FROM EMP  
WHERE DEPT = 10;
```





Les index

CREATE INDEX :

`CREATE [UNIQUE] INDEX nom_index ON table (col1, col2,...);`

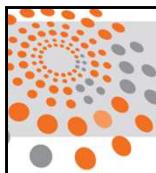
On peut spécifier par l'option UNIQUE que chaque valeur d'index doit être unique dans la table.

DROP INDEX :

`DROP INDEX nom_index [ON table];`

Le nom de la table ne doit être précisé que si vous voulez supprimer un index d'une table d'un autre utilisateur alors que vous possédez un index du même nom.

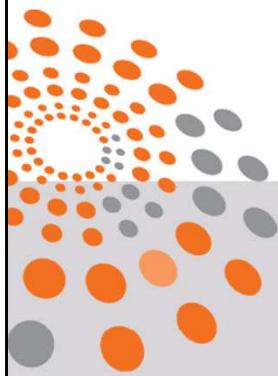




Atelier 7

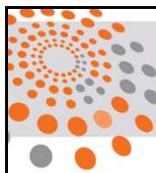
1. Créer une table pays avec 2 champs : code pays (4 caractères, clé primaire), nom pays (40 caractères maximum)
2. Ajouter une colonne courriel (60 caractères possibles) à la table CLIENTS. Puis la modifier pour passer à 75 caractères. Pour finir, supprimer cette colonne.
3. Créez une vue qui affiche le nom de la société, l'adresse, le téléphone et la ville des clients qui habitent en province.





CORRIGÉS DES ATELIERS





Atelier 2.1

1. Affichez tous les employés de la société.
2. Affichez toutes les catégories de produits.
3. Affichez les noms, prénoms et date de naissance, et leur commission (à 0 si pas de commission) de tous les employées de la société.



1.

```
select *  
from EMPLOYES;
```

2.

```
select *  
from CATEGORIES;
```

3.

Dans toutes les bases de données :

```
select NOM , PRENOM , DATE_NAISSANCE, coalesce(COMMISSION,0)  
from EMPLOYES;
```

ORACLE :

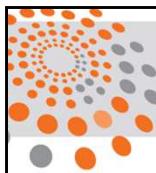
```
select NOM , PRENOM , DATE_NAISSANCE, nvl(COMMISSION,0)  
from EMPLOYES;
```

SQL Server :

```
select NOM , PRENOM , DATE_NAISSANCE, isnull(COMMISSION,0)  
from EMPLOYES;
```

MySQL :

```
select NOM , PRENOM , DATE_NAISSANCE, ifnull(COMMISSION,0)  
from EMPLOYES;
```



Atelier 2.1

4. Affichez la liste des fonctions des employés de la société.
5. Affichez la liste des pays de nos clients.
6. Affichez la liste des localités dans lesquelles il existe au moins un client.



4.

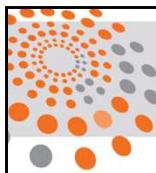
```
select distinct FONCTION  
from EMPLOYES;
```

5.

```
select distinct PAYS  
from CLIENTS;
```

6.

```
select distinct VILLE  
from CLIENTS;
```



Atelier 2.2

1. Affichez les produits commercialisés et la valeur de stock par produit (prix unitaire *quantité en stock).
2. Affichez le nom, le prénom, l'âge et l'ancienneté des employés, dans la société.



1.

```
Select NOM_PRODUIT,  
       PRIX_UNITAIRE*UNITES_STOCK as valeur_stock  
from PRODUITS;
```

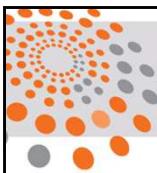
2.

ORACLE :

```
select NOM,  
       PRENOM,  
       round((sysdate-date_naissance)/365) as age,  
       round((sysdate-date_embauche)/365) as ancienneté  
from employes;
```

POSTGRE :

```
select NOM,  
       PRENOM,  
       floor((current_date-date_naissance)/365) as age,  
       floor((current_date-date_embauche)/365) as ancienneté  
from employes;
```



Atelier 2.2

1. Affichez le nom, le prénom, l'âge et l'ancienneté des employés, dans la société.

2. Écrivez la requête qui permet d'afficher :

Employé a un gain annuel sur 12 mois
----- ----- ----- -----
Fuller gagne 120000 par an.
.....



2.

MySQL :

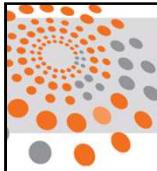
```
select NOM,  
       PRENOM,  
       Datediff(current_date, date_naissance)/365.25 as "AGE",  
       Datediff(current_date, date_embauche)/365.25 as "ANCIENNETE"  
from EMPLOYES;
```

SQL Server :

```
select NOM,  
       PRENOM,  
       datediff(year, DATE_NAISSANCE, getdate()) as "AGE",  
       datediff(year, DATE_EMBAUCHE, getdate()) as "ANCIENNETE"  
from EMPLOYES;
```

3.

```
select NOM as "Employé" ,  
       'gagne' as "a un" ,  
       SALAIRE*12 as "gain annuel",  
       'par an' as "sur 12 mois"  
from EMPLOYES;
```



Atelier 2.3

1. Affichez les nom de la société et le pays des clients qui habitent à Toulouse.
2. Affichez les nom, prénom et fonction des employés dirigés par l'employé numéro 2.
3. Affichez les nom, prénom et fonction des employés qui ne sont pas des représentants.



1.

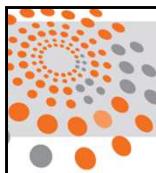
```
select SOCIETE, ADRESSE  
from CLIENTS  
where VILLE ='Toulouse';
```

2.

```
select NOM, PRENOM, FONCTION  
from EMPLOYES  
where REND_COMPTE = 2;
```

3.

```
select NOM,PRENOM,FONCTION  
from EMPLOYES  
where FONCTION <> 'Représentant(e)';
```



Atelier 2.3

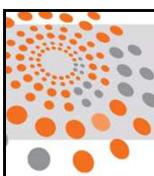
4. Affichez les nom, le prénom et fonction des employés qui ont un salaire inférieur à 3500.
5. Affichez le nom de la société, la ville et le pays des clients qui n'ont pas de fax.
6. Affichez les nom, prénom et la fonction des employés qui n'ont pas de supérieur.



4.
select NOM, PRENOM, FONCTION
from EMPLOYES
where SALAIRE < 3500;

5.
select SOCIETE, VILLE, PAYS
from CLIENTS
where FAX is null;

6.
select NOM, PRENOM, FONCTION
from EMPLOYES
where REND_COMPTE is null;



Atelier 2.4

1. Trier les employés par nom de salarié en ordre décroissant.
2. Trier les clients par pays.
3. Trier les clients par pays et par ville.
4. Trier les employés par commission.



1.select NOM, PRENOM from EMPLOYES order by nom desc;

2.

```
select SOCIETE, VILLE  
from CLIENTS  
order by PAYS;
```

3.

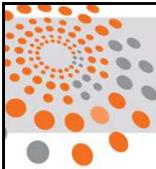
```
select SOCIETE, VILLE, PAYS  
from CLIENTS  
order by PAYS, VILLE;
```

4.

```
select NOM, COMMISSION  
from EMPLOYES  
order by COMMISSION;
```

ORACLE et POSTGRE :

```
select NOM, COMMISSION  
from EMPLOYES  
order by COMMISSION nulls first;
```



Atelier 2.5

1. Affichez le nom, prénom, fonction et salaire des employés qui ont un salaire compris entre 2500 et 3500.
2. Affichez le nom du produit, le nom du fournisseur, le nom de la catégorie et les quantités de produits qui ne sont pas d'une des catégories 1,3,5 et 7.
3. Affichez le nom du produit, le nom du fournisseur, le nom de la catégorie et les quantités des produits qui ont le numéro fournisseur entre 1 et 3 ou un code catégorie entre 1 et 3, et pour lesquelles les quantités sont données en boîtes ou en cartons.



1.

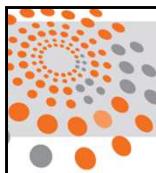
```
select NOM, PRENOM, FONCTION  
from EMPLOYES  
where SALAIRE between 2500 and 3500;
```

2.

```
select NOM_PRODUIT, QUANTITE, SOCIETE, NOM_CATEGORIE  
from CATEGORIES cat  
join PRODUITS p on cat. CODE_CATEGORIE=p.CODE_CATEGORIE  
join FOURNISSEURS f on p.NO_FOURNISSEUR=f.NO_FOURNISSEUR  
where p.CODE_CATEGORIE not in (1,3,5,7);
```

3.

```
select NOM_PRODUIT, QUANTITE, SOCIETE, NOM_CATEGORIE  
from CATEGORIES cat  
join PRODUITS p on cat. CODE_CATEGORIE=p.CODE_CATEGORIE  
join FOURNISSEURS f on p.NO_FOURNISSEUR=f.NO_FOURNISSEUR  
where (p.NO_FOURNISSEUR between 1 and 3  
or p.CODE_CATEGORIE between 1 and 3)  
and (QUANTITE like '%boîtes%' or QUANTITE like '%cartons%');
```



Atelier 2.5

4. Écrivez la requête qui permet d'afficher le nom des employés qui ont effectué au moins une vente pour un client parisien.
5. Affichez le nom des produits et le nom des fournisseurs pour les produits des catégories 1,4 et 7.
6. Affichez la liste des employés ainsi que le nom de leur supérieur hiérarchique.



4.

```
select NOM,SOCIETE,VILLE  
from EMPLOYES e  
join COMMANDES cmd on e.NO_EMPLOYEE=cmd.NO_EMPLOYEE  
join CLIENTS cl on cmd.CODE_CLIENT=cl.CODE_CLIENT  
where VILLE='Paris';
```

5.

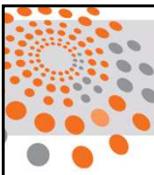
```
select NOM_PRODUIT, SOCIETE from PRODUITS p  
join FOURNISSEURS f on p.NO_FOURNISSEUR=f.NO_FOURNISSEUR  
where CODE_CATEGORIE in (1,4,7);
```

6.

```
select emp.NOM as "Employé",  
       chef.NOM as "Supérieur"  
from EMPLOYES emp  
join EMPLOYES chef  
on a.REND_COMPTE = b.no_employe;
```

OU, si on souhaite afficher tous les employés :

```
select emp.NOM as "Employé",  
       chef.NOM as "Supérieur" from  
EMPLOYES emp  
left join EMPLOYES chef  
on emp.REND_COMPTE = chef.NO_EMPLOYEE;
```



Atelier 3.1

1. Affichez la somme des salaires et des commissions des employés en gérant les valeurs null.
2. Affichez la moyenne des salaires et des commissions des employés.



1.

```
select sum(SALAIRE) +sum(COMMISSION) as "Total revenus"  
from EMPLOYES;
```

2.

Toutes bases de données :

```
select round(avg(SALAIRE),2) as "moyenne salaire",  
       round(avg(coalesce(COMMISSION,0)),2) as "moyenne commission"  
from EMPLOYES;
```

ORACLE :

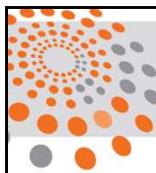
```
select round(avg(SALAIRE),2) as "moyenne salaire",  
       round(avg(nvl(COMMISSION,0)),2) as "moyenne commission"  
from employees;
```

SQL Server

```
select round(avg(SALAIRE),2) as "moyenne salaire",  
       round(avg(isnull(COMMISSION,0)),2) as "moyenne commission"  
from EMPLOYES;
```

MySQL

```
select round(avg(SALAIRE),2) as "moyenne salaire",  
       round(avg(ifnull(COMMISSION,0)),2) as "moyenne commission"  
from EMPLOYES;
```



Atelier 3.1

3. Affichez le salaire maximum et la plus petite commission des employés.
4. Affichez le nombre distinct de fonction.

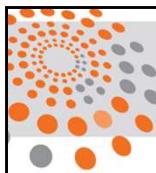


3.

```
select max(SALAIRE) as "plus grand salaire",
       min(COMMISSION) as "plus petite commission"
  from EMPLOYES;
```

4.

```
select count( distinct FONCTION)
  from EMPLOYES;
```



Atelier 3.2

1. Écrivez la requête qui permet d'afficher la masse salariale des employés par fonction.
2. Affichez le total des commandes, pour les commandes qui comportent plus de 5 références de produit.
3. Afficher la valeur des produits en stock et la valeur des produits commandés par fournisseur, pour les fournisseurs qui ont un numéro compris entre 3 et 6.



1.

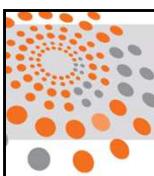
```
select FONCTION , sum(SALAIRE) as "MASSE SALARIALE"  
from EMPLOYES  
group by FONCTION;
```

2.

```
select NO_COMMANDE,  
       sum( PRIX_UNITAIRE * QUANTITE ) as "TOTAL"  
  from DETAILS_COMMANDES  
group by NO_COMMANDE  
having count(NO_COMMANDE) > 5;
```

3.

```
select NO_FOURNISSEUR,  
       sum( PRIX_UNITAIRE * UNITES_STOCK ) as "EN STOCK",  
       sum( PRIX_UNITAIRE * UNITES_COMMANDEES ) as "VENDU"  
  from PRODUITS  
 where NO_FOURNISSEUR BETWEEN 3 and 6  
group by NO_FOURNISSEUR;
```



Atelier 4

1. Affichez les sociétés, adresse et villes de résidence pour tous les tiers de l'entreprise.
2. Affichez toutes les commandes qui comportent en même temps des produits de catégorie 1 du fournisseur 1 et des produits de catégorie 2 du fournisseur 2.



1.

```
select SOCIETE, ADRESSE, VILLE from CLIENTS union
select SOCIETE, ADRESSE, VILLE from FOURNISSEURS;
```

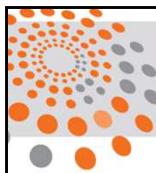
2.

ORACLE et SQL SERVER et POSTGRE

```
select NO_COMMANDE from DETAILS_COMMANDES dc
join PRODUITS p on dc.REF_PRODUIT=p.REF_PRODUIT
where CODE_CATEGORIE=1 and NO_FOURNISSEUR=1
intersect
select NO_COMMANDE from DETAILS_COMMANDES dc
join PRODUITS p on dc.REF_PRODUIT=p.REF_PRODUIT
where CODE_CATEGORIE=2 and NO_FOURNISSEUR=2;
```

Toutes bases de données :

```
select NO_COMMANDE from DETAILS_COMMANDES dc
join PRODUITS p on dc.REF_PRODUIT=p.REF_PRODUIT
where CODE_CATEGORIE=1 and NO_FOURNISSEUR=1
and no_commande not in (
    select NO_COMMANDE
        from DETAILS_COMMANDES dc
        join PRODUITS p
            on dc.REF_PRODUIT=p.REF_PRODUIT
        where CODE_CATEGORIE=2
        and NO_FOURNISSEUR=2);
```



Atelier 4

3. Affichez la liste des produits que les clients parisiens ne commandent pas.



3.ORACLE

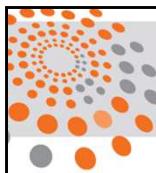
```
select REF_PRODUIT from PRODUITS
minus
select REF_PRODUIT
from DETAILS_COMMANDES dc
join COMMANDES cmd on dc.NO_COMMANDE=cmd.NO_COMMANDE
join CLIENTS cl on cmd.CODE_CLIENT=cl.CODE_CLIENT
where VILLE='Paris';
```

SQL SERVER et POSTGRE

```
select REF_PRODUIT from PRODUITS
except
select REF_PRODUIT
from DETAILS_COMMANDES dc
join COMMANDES cmd on dc.NO_COMMANDE=cmd.NO_COMMANDE
join CLIENTS cl on cmd.CODE_CLIENT=cl.CODE_CLIENT
where VILLE='Paris';
```

Toutes bases de données

```
select REF_PRODUIT from PRODUITS
where REF_PRODUIT not in (select REF_PRODUIT
                           from DETAILS_COMMANDES dc
                           join COMMANDES cmd
                           on dc.NO_COMMANDE=cmd.NO_COMMANDE
                           join CLIENTS cl
                           on cmd.CODE_CLIENT=cl.CODE_CLIENT
                           where VILLE='Paris');
```



Atelier 5

1. Affichez tous les produits pour lesquels la quantité en stock est inférieur à la moyenne des quantités en stock.
2. Affichez toutes les commandes pour lesquelles les frais de ports dépassent la moyenne des frais de ports pour ce client.
3. Affichez les produits pour lesquels la quantité en stock est supérieure à la quantité en stock de chacun des produits de catégorie 3.



1.

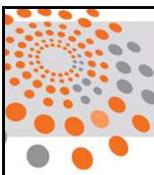
```
select NOM_PRODUIT from PRODUITS  
where UNITES_STOCK < ( select avg(UNITES_STOCK) from PRODUITS);
```

2.

```
select CODE_CLIENT, NO_COMMANDE, PORT  
from COMMANDES c  
where PORT > ( select avg(PORT) as moyenne  
                  from COMMANDES b  
                  where c.CODE_CLIENT=b.CODE_CLIENT);
```

3.

```
select REF_PRODUIT from PRODUITS  
where UNITES_STOCK > (select max(UNITES_STOCK)  
                        from PRODUITS  
                        where CODE_CATEGORIE = 3);
```

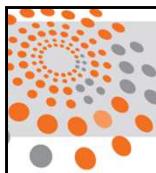


Atelier 6

1. Insérez une nouvelle catégorie de produits nommée «fruits et légumes », en respectant les contraintes.
2. Créez un nouveau fournisseur « Grandma » (no_fournisseur = 30) avec les mêmes coordonnées que le fournisseur « Grandma Kelly's Homestead ».
3. Attribuer les produits de « Grandma Kelly's Homestead » au nouveau fournisseur créé.
4. Supprimez l'ancien fournisseur « Grandma Kelly's Homestead ».



1.
insert into CATEGORIES (CODE_CATEGORIE, NOM_CATEGORIE, DESCRIPTION)
values (9, 'Fruits et Légumes', 'Fruits frais et confits, légumes de saison');
2.
insert into FOURNISSEURS
select 33,SOCIETE, ADRESSE, VILLE, CODE_POSTAL, PAYS, TELEPHONE,FAX
from FOURNISSEURS
where SOCIETE = 'Grandma Kelly"s Homestead';
3.
update PRODUITS
set NO_FOURNISSEUR=33
where NO_FOURNISSEUR=3
4.
delete from FOURNISSEURS
where NO_FOURNISSEUR = (select min(NO_FOURNISSEUR)
from FOURNISSEURS
where SOCIETE like 'Grandma Kelly"s Homestead ');



Atelier 7

1. Créer une table pays avec 2 champs : code pays (4 caractères, clé primaire), nom pays (40 caractères maximum).
2. Ajouter une colonne courriel (75 caractères possibles) à la table CLIENTS. Puis la modifier pour passer à 60 caractères. Pour finir, supprimer cette colonne.
3. Créez une vue qui affiche le nom de la société, l'adresse, le téléphone et la ville des clients qui habitent en province.



1.

```
create table pays  
(code_pays char(4) not null primary key,  
nom_pays varchar(40)  
);
```

2.

Ajout d'une colonne
alter table CLIENTS add (courriel varchar2(75));

Modification d'une colonne
ORACLE et MySQL

```
alter table CLIENTS modify (courriel varchar2(60));
```

SQL Server et Postgre
alter table CLIENTS alter column (courriel varchar2(60));

Suppression d'une colonne
alter table CLIENTS drop column courriel;

3.

```
create view V_CLIENTS_PROVINCE as  
select SOCIETE, ADRESSE, TELEPHONE, VILLE  
from CLIENTS  
where PAYS='France'  
and CODE_POSTAL not like '75%' and CODE_POSTAL not like '77%',  
and CODE_POSTAL not like '78%' and CODE_POSTAL not like '91%',  
and CODE_POSTAL not like '92%' and CODE_POSTAL not like '93%',  
and CODE_POSTAL not like '94%' and CODE_POSTAL not like '95%'  
;
```