Introduction aux bases de donnés

Les fonctions de groupement.

Fonctions de groupements

Plan de la séance

- Retour sur la dernière séance:
  - Point de vue de l'étudiant
  - Point de vue de l'enseignant.
- Rappels:
  - La commande SELECT et jointures
- Les fonctions:
  - MAX, MIN
  - AVG,SUM
  - COUNT
- La clause GROUP BY
- La clause Having

Définition et exemples

Définition

- Les fonctions de groupement sont des fonctions utilisées pour traiter des groupes de rangées(lignes) et d'afficher un seul résultat.
- Exemples:
  - Nous souhaitons connaître le salaire moyen des employés du département d'informatique
  - Nous souhaitons connaître la masse salariale (somme des salaires de tous les joueurs) du canadien de Montréal
  - Le nombre total d'étudiants au collège Lionel Groulx
  - Le nombre d'étudiants dans chaque programme au collège Lionel Groulx.

Les fonction MIN et MAX

La fonction MIN

Cette fonction permet de chercher le **MINIMUM** parmi l'ensemble des valeurs de la colonne indiquée.

Syntaxe simplifiée

```
SELECT MIN(colonne) FROM nomTable [WHERE expression]
```

Exemple

```
SELECT MIN(salaire) FROM Employes
```

La fonction MAX

Cette fonction permet de chercher le **MAXIMUM** parmi l'ensemble des valeurs de la colonne indiquée.

Syntaxe simplifiée

```
SELECT MAX(colonne) FROM nomTable [WHERE expression]
```

Exemple

```
SELECT MAX(salaire) FROM Employes
```

Les fonctions AVG et SUM

La fonction AVG

Cette fonction permet de chercher la MOYENNE de l'ensemble des valeurs de la colonne indiquée.

Syntaxe simplifiée SELECT AVG(colonne) FROM nomTable [WHERE expression]

Exemple SELECT AVG(salaire) FROM Employes WHERE deptno =10

La fonction SUM

Cette fonction permet de chercher La SOMME de toutes des valeurs de la colonne indiquée.

Syntaxe simplifiée SELECT SUM(colonne) FROM nomTable [WHERE expression]

Exemple SELECT SUM(salaire) FROM Employes WHERE deptno =10

La fonction COUNT

La fonction COUNT

Cette fonction permet de compter le nombre de lignes (rangées) qui répondent à un critère.

Syntaxe simplifiée

```
SELECT COUNT ( {*| [DISTINCT |ALL] nomcolonne})  FROM nomTable [WHERE expression]
```

- count(*) On ramène le nombre total, même ceux ayant des colonnes avec des valeurs nulles
- count(colonne) :Dans ce cas on calcule le nombre lignes dans la table selon la colonne indiquée. Seules les lignes avec des valeurs de la colonne indiquée qui sont NOT NULL seront comptées.
- count(ALL colonne) :Dans ce cas on calcule le nombre lignes dans la table selon la colonne indiquée. Les lignes ayant des valeurs nulles pour la colonnes ne seront pas comptées
- count(DISTINCT colonne) :Dans ce cas on calcule le nombre lignes dans la table selon la colonne indiquée. Seules les lignes avec des valeurs de la colonne indiquée qui sont NOT NULL seront comptées. De plus les lignes avec des valeurs identiques de la colonnes indiquées seront comptées une seule fois.

Voir l'exemple de l'acétate après pour mieux comprendre

La fonction COUNT

La fonction COUNT

- **SELECT COUNT (*) FROM joueurs** va retourner 12(on compte tout le monde)
- **SELECT COUNT (numjoueur) FROM joueurs** va retourner 12. car numjoueur est la PK il est donc NOT NULL. Il y a 12 numjoueur
- **SELECT COUNT(nom) FROM Joueurs** va retourner 10 car il y a 10 joueurs avec un nom. (2 ont NULL à nom)
- **SELECT COUNT(ALL nom) FROM Joueurs** va retourner 10 on compte tous les noms. Il y a 10 nom
- **SELECT COUNT (DISTINCT nom) FROM Joueurs** va retourner 9 car on compte que les noms DISTINCT (différents)

Il est recommandé d'utiliser le * lorsque vous voulez compter TOUTES les lignes.

	NUMJOUEUR	NOM	PRENOM	CODEEQUIPE	SALAIRE
1	1	PRICE	CAREY	MTL	1999999
2	2	MARKOV	ANDRÉ	MTL	1546357
3	3	SUBBAN	KARL	MTL	1654657
4	4	PATIORETTY	MAX	MTL	500000
5	10	HAMOND	ANDREW	OTT	1234565
6	6	STONE	MARC	OTT	1234567
7	9	TURIS	KYLE	OTT	870697
8	7	GALLAGHER	BRANDON	MTL	534543
9	8	TANGUAY	ALEX	AVL	1543456
10	11	PRICE	BIL	AVL	798098
11	50	(null)	Leprenom	(null)	(null)
12	52	(null)	Leprenom2	(null)	(null)

La Clause GROUP BY

La clause GROUP BY

- Parfois, il est nécessaire de grouper les enregistrements avant de les compter. Par exemple dans la table joueurs ci-après, comment déterminer le nombre de joueurs dans chaque équipe. ?

Pour répondre à la question, il suffira de grouper les joueurs selon leurs codeequipe, puis les compter. Rien de plus simple.

- Pour grouper des enregistrements on utilise la clause GROUP BY.

	NUMJOUEUR	NOM	PRENOM	CODEEQUIPE	SALAIRE
1	1	PRICE	CAREY	MTL	1999999
2	2	MARKOV	ANDRÉ	MTL	1546357
3	3	SUBBAN	KARL	MTL	1654657
4	4	PATIORETTY	MAX	MTL	500000
5	10	HAMOND	ANDREW	OTT	1234565
6	6	STONE	MARC	OTT	1234567
7	9	TURIS	KYLE	OTT	870697
8	7	GALLAGHER	BRANDON	MTL	534543
9	8	TANGUAY	ALEX	AVL	1543456
10	11	PRICE	BIL	AVL	798098
11	50	(null)	Leprenom	(null)	(null)
12	52	(null)	Leprenom2	(null)	(null)

La Clause GROUP BY

La clause GROUP BY, exemple de la table joueurs précédentes

```
SELECT COUNT(*), codeequipe
FROM joueurs
GROUP BY codeequipe ;
```

a comme résultat

	COUNT(*)	CODEE...
1	2	AVL
2	2	(null)
3	3	OTT
4	5	MTL

Très important pour la clause GROUP BY.

Toutes les colonnes qui apparaissent dans le SELECT doivent apparaitre dans la clause GROUP BY

Exemple, la requête suivante va renvoyer une erreur (*ORA-00979: n'est pas une expression GROUP BY*) car salaire n'est pas dans le GROUP BY alors qu'il est dans SELECT

```
SELECT COUNT(*), codeequipe, salaire FROM joueurs
GROUP BY codeequipe ;
```

La Clause GROUP BY

Autre Exemple:

Pour raffiner la requête, on pourrait utiliser un ALIAS et ordonner le résultat de la requête

```
SELECT COUNT(*),  AS nbJoueurs , codeequipe
FROM joueurs
GROUP BY codeequipe
ORDER BY nbJoueurs DESC;
```

	NBJOUEURS	CODEEQUIPE
1	5	MTL
2	3	OTT
3	2	AVL
4	2	(null)

La Clause HAVING

La clause HAVING:

Parfois, il nécessaire de cibler ou de restreindre les enregistrements spécifiés. Comme par exemple, ne sortir que les codeequipe ayant le nombre de joueurs supérieurs ou égal à 3. Dans ce cas on utilise la clause HAVING.

La clause HAVING permet de mieux cibler les enregistrements spécifiés. Cette clause s'utilise à la place du WHERE une fois que le GROUP BY est réalisé.

```
SELECT COUNT(*) AS nbJoueurs , codeequipe
FROM joueurs
GROUP BY codeequipe
HAVING COUNT(*)>=3
ORDER BY nbJoueurs DESC;
```

Après la clause GROUP BY, un WHERE n'est plus possible. Il faut utiliser HAVING

Dans la clause HAVING , nous ne pouvons pas utiliser l'ALIAS

La Clause HAVING

La clause HAVING:

La requête suivante va renvoyer une erreur *ORA-00904: « nbJoueurs" : identificateur non valide* car nbJoueurs ne doit pas s'utiliser dans la clause HAVING

```
SELECT COUNT(*) AS nbJoueurs , codeequipe
FROM joueurs
GROUP BY codeequipe
HAVING nbJoueurs >=3
ORDER BY nbJoueurs DESC;
```

Les Clauses HAVING et WHERE

La clause WHERE peut-être utilisée dans une requête de groupement, il faudra l'utiliser avant le GROUP BY

La clause HAVING peut-être utilisée dans une requête de groupement, il faudra l'utiliser après le GROUP BY

Lorsque c'est possible, il vaut mieux utiliser un WHERE à la place du HAVING . La requête dans le carrée vert est mieux (Car on réduit les lignes à grouper) que la requête dans le carré gris. Les deux requêtes donnent le même résultats.

```
SELECT COUNT(*) AS nbJoueur,CODEEQUIPE
FROM JOUEURS  WHERE codeequipe ='MTL'
GROUP BY codeequipe;
```

```
SELECT COUNT(*) AS nbJoueur,CODEEQUIPE
FROM JOUEURS
GROUP BY codeequipe
HAVING  codeequipe ='MTL';
```

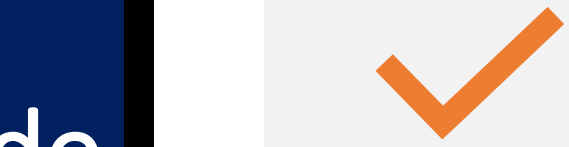
Les Clauses GROUP BY et HAVING

- Les clauses GROUP BY ET HAVING s'utilisent également sur les fonctions MIN, MAX, AVG et SUM
- Les fonctions de groupements s'utilisent aussi avec des jointures

```
SELECT COUNT(*) AS nbJoueurs,nomEquipe
FROM ( Joueurs INNER JOIN  equipes ON joueurs.codeequipe =equipes.codeequipe)
GROUP BY equipes.NOMEQUIPE
ORDER BY nbJoueurs DESC;
```

```
SELECT AVG(Salaire) AS MoyenneSalaire,nomEquipe
FROM (Joueurs inner join equipes on joueurs.codeequipe =equipes.codeequipe)
GROUP BY equipes.NOMEQUIPE
HAVING AVG(salaire)>1200000;
```

Les fonctions de groupement



Conclusion

Questions