# 2ème Partie : Les Fonctions Mono lignes

**Objectifs de la partie:**

Définir et savoir utiliser les :

i. Fonctions de caractères
ii. Fonctions de manipulation des caractères
iii. Fonctions de manipulation des dates
iv. Fonctions numériques
v. Fonctions de conversions
vi. Autres fonctions

# Fonctions de caractères. Conversion Minuscule/Majuscule(1/2)

**Il existe trois fonctions principales:**

- LOWER : convertir les caractères « M » en « m »
- UPPER : convertir les caractères « m » en « M »
- INITCAP : convertir l'initiale de chaque mot en « M » et les caractères suivants en « m »

- **Remarque:** Une fonction mono ligne est une fonction qui s'applique enregistrement par enregistrement.

# Fonctions de caractères. Conversion Minuscule/Majuscule(2/2)

SELECT LOWER(last_name) as "prénom" FROM employees;

| Prénom |
| --- |
| abel |
| ande |

SELECT UPPER(last_name) as "prénom" FROM employees;

| Prénom |
| --- |
| ABEL |
| ANDE |

SELECT INITCAP(last_name) as "prénom" FROM employees;

| Prénom |
| --- |
| Abel |
| Ande |

# Fonctions de manipulation des caractères. Définitions (1/2)

- CONCAT(chaine1,chaine2) : concatène 2 chaines = ||
- SUBSTR(chaine, pos, taille) : extrait une sous chaîne d'une autre chaîne
- LENGTH(ch) : taille d'une chaîne en caractères
- INSTR(ch,sch) : position d'une chaîne de caractères dans une autre chaîne
- TRIM(ch) : élimine les espaces à gauche et à droite
- LTRIM(ch) : élimine les espaces à gauche
- RTRIM(ch) : élimine les espaces à droite

# Fonctions de manipulation des caractères. Définitions (2/2)

- LPAD (chaine, nbr, carac) : complète une chaîne de caractères sur la gauche avec une autre chaîne pour avoir n caractères.
- RPAD (chaine, nbr, carac): complète une chaîne de caractères sur la droite avec une autre chaîne pour avoir n caractères.
- ASCII (chaine) : retourne le code ascii du premier caractère de la chaîne.
- CHR (nbr): retourne le caractère (inverse de ascii).

# Fonctions de manipulation des caractères. Exemples

```
SELECT    'station' ch1, 'agil' ch2,
          Concat ('station 1','agil') "la station",
          Substr ('station 1',9,1) "numéro station",
          Length ('agil') "longueur ch1",
          Instr ('agil','i') "position de i",
          Length (Trim('   agil   ')) "trim",
          Length (Rtrim('   agil   ')) "Rtrim",
          Length (Ltrim('   agil   ')) "Ltrim",
          Rpad ('agil',8,'*') "RPAD", Lpad ('agil',8,'-') "LPAD",
          ASCII ('test') " code ascii " , CHR(75)
FROM   Dual;
```

# Fonctions numériques. Présentation

- **ROUND** (x, n) : Arrondir la valeur de x à la précision spécifiée n
- **TRUNC** (x, n) : Tronquer la valeur de x à la précision spécifiée n
- **FLOOR** (x) : si n<x<n+1 alors FLOOR(x)=n
- **CEIL** (x) : si n<x<n+1 alors CEIL(x)=n+1
- **MOD** (x, y) : reste de la division de x sur y
- **REMAINDER** (m, n) : reste d'une division calculé comme suit : m-(q*n) avec q égale à la partie entière du quotient

# Fonctions numériques. Round (1/2)

```
SELECT   round(123.646, 0) "0 chiffres" , round(123.646, 1) "1 chiffres",
         round(123.646, 2) "2 chiffres", round(123.646, 3) "3 chiffres"
FROM Dual;
```

**Résultat:**

| 0 Chiffres | 1 Chiffres | 2 Chiffres | 3 Chiffres |
|---|---|---|---|
| 124 | 123.6 | 123.65 | 123.646 |

# Fonctions numériques. Round (2/2)

```
SELECT   round(123.646, -1) "1 chiffres" ,round(123.646, -2) "2 chiffres",
                round(123.646, -3) "3 chiffres"
FROM Dual;
```

**Résultat:**

| 1 Chiffres | 2 Chiffres | 3 Chiffres |
|---|---|---|
| 120 | 100 | 0 |

# Fonctions numériques. TRUNC

```
SELECT      TRUNC (123.646, 0) "n=0", TRUNC (123.646, 1) "n=1",
 TRUNC (123.646, 2) "n=2", TRUNC (123.646, 3) "n=3", TRUNC (123.636, -1)
  "n=-1", TRUNC (123.646, -2) "n=-2", TRUNC (123.646, -3) "n=-3"
FROM Dual;
```

**Résultat:**

| N=0 | N=1 | N=2 | N=3 | N=-1 | N=-2 | N=-3 |
|---|---|---|---|---|---|---|
| 123 | 123.6 | 123.64 | 123.646 | 120 | 100 | 0 |

# Fonctions numériques. FLOOR/ CEIL

```
SELECT      floor(-2.56), floor(-2.02), floor(-2.8),
            ceil(-2.56), ceil(-2.02), ceil(-2.8)
FROM dual;
```

**Résultat:**

| FLOOR(-2.56) | FLOOR(-2.02) | FLOOR(-2.8) | CEIL(-2.56) | CEIL(-2.02) | CEIL(-2.8) |
|---|---|---|---|---|---|
| -3 | -3 | -3 | -2 | -2 | -2 |

```
SELECT      floor(2.56), floor(2.02), floor(2.8),
            ceil(2.56), ceil(2.02), ceil(2.8)
FROM dual;
```

**Résultat:**

| FLOOR(2.56) | FLOOR(2.02) | FLOOR(2.8) | CEIL(2.56) | CEIL(2.02) | CEIL(2.8) |
|---|---|---|---|---|---|
| 2 | 2 | 2 | 3 | 3 | 3 |

# Fonctions numériques. MOD/ REMAINDER

```
SELECT    MOD (12, 4), MOD (16, 6)
FROM Dual;
```

**Résultat:**

| MOD(12,4) | MOD(16,6) |
|---|---|
| 0 | 4 |

```
SELECT    remainder (15, 6),  remainder (15, 5),
          remainder (15, 4),  remainder (-15,4)
FROM Dual;
```

15-(2*6) = **3** / 15-(3*5) = **0** / 15-(4*4) = **-1** / -15-(-4*4) = **1**

**Résultat:**

| REMAINDER(15,6) | REMAINDER(15,5) | REMAINDER(15,4) | REMAINDER(-15,4) |
|---|---|---|---|
| 3 | 0 | -1 | 1 |

# Fonctions de manipulation des dates. (1/3)

- **MONTHS_BETWEEN (date1,date2):** nombre de mois entre deux dates
- **ADD_MONTHS (date, nb_mois):** Ajoute des mois calendaires à une date
- **NEXT_DAY (date, jour)** : le jour suivant
- **LAST_DAY (date) :** le dernier jour du mois
- **ROUND (date, précision) :** arrondi une date
- **TRUNC (date, précision) :** tronque une date
- **EXTRACT (day/month/year/hour/minute/seconde from date):** extraction du jour, mois, année, heure, minute et seconde

# Fonctions de manipulation des dates. (2/3)

```
SELECT    sysdate "date du jour",
          sysdate+2 "Date jr plus 2jrs",
          to_date ('10/09/2012','dd/mm/yyyy')-10 "date du jour moins 10" ,
          Trunc (sysdate - to_date('01/05/2012','dd/mm/yyyy'),0)
          "tronquer une date"
FROM dual ;
```

**Résultat:**

| Date Du Jour | Date Jr Plus 2jrs | Date Du Jour Moins 10 | Tronquer Une Date |
|---|---|---|---|
| 09-MAY-12 | 11-MAY-12 | 31-AUG-12 | 8 |

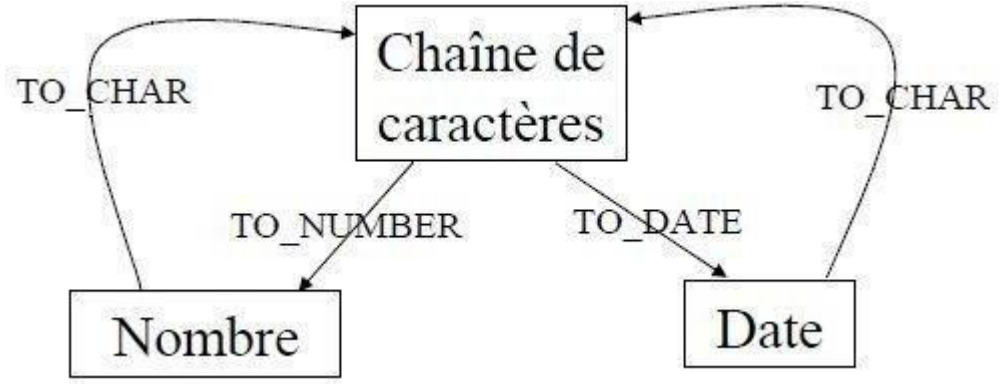
# Fonctions de manipulation des dates. (3/3)

```
SELECT
Trunc (months_between (to_date ('2012/01/01', 'yyyy/mm/dd'), to_date
('2012/03/15', 'yyyy/mm/dd') ),2) "nbr de mois",
next_day(to_date ('2012/03/01', 'yyyy/mm/dd'), 'Monday') "lundi suivant",
last_day(to_date ('2012/03/01', 'yyyy/mm/dd')) "dernier jr du mois mars12" ,
add_months(to_date ('2012/03/01', 'yyyy/mm/dd'),3) "ajouter 3 mois"
FROM Dual;
```

**Résultat:**

| Nbr De Mois | Lundi Suivant | Dernier Jr Du Mois Mars12 | Ajouter 3 Mois |
|---|---|---|---|
| -2.45 | 05-MAR-12 | 31-MAR-12 | 01-JUN-12 |

# Fonctions de conversions.




- **TO_DATE :** Convertir une Chaine en format Date
- **TO_NUMBER :** Convertir une Chaine en format Numérique
- **TO_CHAR :** Convertir une Date /un nombre en une Chaine

# Fonctions de conversions.

## Options avec To_char

| OPTION | DESCRIPTION |
|---|---|
| yyyy | l'année (quatre chiffres) |
| month | nom complet du mois en minuscule |
| mon | abréviation du nom du mois en minuscule (trois caractères) |
| day | nom complet du jour en minuscule |
| DD | jour du mois (01-31) |
| D | jour de la semaine (de 1 à 7, dimanche étant le 1) |
| dy | abréviation du nom du jour en minuscule (3 caractères) |
| ddd | jour de l'année (001-366) |
| q | trimestre |
| w | numéro de semaine du mois (de 1 à 5) (la première semaine commence le premier jour du mois.) |
| ww | numéro de semaine dans l'année (de 1 à 53) (la première semaine commence le premier jour de l'année. |

# Autres Fonctions.



- **NVL (nom_col, valeur) :** remplace une valeur nulle
- **NVL2 (expr,val1,val2) :** si expr n'est pas nulle alors elle est remplacée par val1 sinon par val2.
- **NULLIF (val1,val2) :** si val1= val2 la valeur NULL est retournée sinon val1
- **Case :** évalue une liste de conditions et retourne un résultat parmi les cas possibles

# Autres Fonctions. (2/5)

- Ecrire les requêtes SELECT qui permettent de:
  - Remplacer « commission_pct » par 0 au niveau de la table « employees » si elle est null
  - Remplacer « commission_pct » par 0 au niveau de la table « employees » si elle est null sinon par 1
  - Afficher la liste des employés en ajoutant une colonne « nom_departement » qui affiche :
    - Si department_id=10 , nom_departement='depart_10'
    - Si department_id=20 , nom_departement='depart_20'
    - Si department_id=30 , nom_departement='depart_30'
    - …

# Autres Fonctions. (3/5)

- **ROW_NUMBER :** retourne le numéro séquentiel d'une ligne d'une partition d'un ensemble de résultats, en commençant à 1 pour la première ligne de chaque partition. Ne prend pas en considération les doublons

- **RANK:** retourne le rang de chaque ligne au sein de la partition d'un ensemble de résultats. Compte les doublons mais laisse des trous

- **DENSE_RANK :** retourne le rang des lignes à l'intérieur de la partition d'un ensemble de résultats, sans aucun vide dans le classement.

  Compte les doublons mais ne laisse pas des trous

# Autres Fonctions.



```
SELECT
    Row_number() over  (partition by <col x> order by <col_y>
    DESC/ASC),  Col1,...colN
FROM <nom_table> ;
```

**A retenir:**

- La clause <order by> est obligatoire.
- La clause <partition by> est facultative, est utilisée pour faire un ordre par ensemble de lignes selon <colx>.
- Rank et dense_rank ont la même syntaxe.

# Autres Fonctions. (5/5)

Ecrire les requêtes SELECT qui permettent de :

- Afficher la liste des employees numérotés par « department_id ».
- Afficher la liste des employees numérotés par département et selon le salaire décroissant :
  - Utiliser « row_number »
  - Utiliser « rank »
  - Utiliser « dense_rank »

# 3ème Partie :

## Les Fonctions Multi lignes ( ou fonctions de groupe )

**Objectif du cours :**

Définir la clause « GROUP BY »,

la clause « HAVING »

# Fonctions Multi lignes. Définitions (1/3)

Une fonction mono ligne est une fonction qui s'applique enregistrement par enregistrement.
Une fonction multi ligne ou fonction de groupe s'applique sur un groupe d'enregistrements et donne un résultat par groupe.

- Clause GROUP BY : Définit le critère de groupement pour la fonction
- Clause HAVING : Permet de mettre des conditions sur les groupes d'enregistrements

# Fonctions Multi lignes. Définitions (2/3)

- COUNT : Nombre de lignes,
- SUM : Somme des valeurs,
- AVG : Moyenne des valeurs,
- MIN : Minimum,
- MAX : Maximum,
- STDDEV: Ecart type,
- VARIANCE: Variance

# Fonctions Multi lignes. Exemple1

```
SELECT      COUNT (salary) as count,
            Trunc (SUM (salary),2) as sum,
            MIN (salary) as min,
            MAX (salary) as max,
            Trunc (AVG (salary),2) as avg,
            Trunc (VARIANCE (salary),2) as variance,
            Trunc (STDDEV(salary),2) as ecart
FROM employees;
```

**Résultat:**

| COUNT | SUM | MIN | MAX | AVG | VARIANCE | ECART |
|---|---|---|---|---|---|---|
| 107 | 691400 | 2100 | 24000 | 6461,682 | 15283140,539 | 3909,365 |

# Fonctions Multi lignes. Syntaxe de la clause GROUP BY

```
SELECT <colonnes>, <fonction de groupe>
FROM table
WHERE <conditions>
GROUP BY <col>|<expr>
HAVING <conditions>
ORDER BY <col>|<expr>
```

# Fonctions Multi lignes. Exemple2

```
SELECT department_id,
COUNT(employee_id) "nbr employes"
FROM employees
GROUP BY department_id;
```

| DEPARTMENT_ID | Nbr Employes |
|---|---|
| 100 | 6 |
| 30 | 6 |
| - | 1 |
| 90 | 3 |
| 20 | 2 |
| 70 | 1 |
| 110 | 2 |
| 50 | 45 |
| 80 | 34 |
| 40 | 1 |
| 60 | 5 |
| 10 | 1 |

**12 Départements**

```
SELECT department_id, count(employee_id) "nbr employes"
FROM employees
GROUP BY department_id
HAVING department_id<40;
```

| DEPARTMENT_ID | Nbr Employes |
|---|---|
| 30 | 6 |
| 20 | 2 |
| 10 | 1 |