

## AGGREGATING DATA USING GROUP FUNCTIONS

### -- CHAPTER 5 --

- What are group functions? These are functions that operate on sets of rows to give one result per group.
  - **AVG** ([DISTINCT] | ALL] n) :
    - Average value of n, ignoring null values.
  - **COUNT** ({\* | [DISTINCT] | ALL] expr) :
    - Returns the number of rows where expr evaluates to something other than null.
  - **MAX** ([DISTINCT] | ALL] expr) :
    - Returns the largest value ignoring all null values.
  - **MIN** ([DISTINCT] | ALL] expr) :
    - Returns the smallest value ignoring all null values.
  - **STDDEV** ([DISTINCT] | ALL] x) :
    - Standard deviation of n, ignoring null values.
  - **SUM** ([DISTINCT] | ALL | n) :
    - Returns the sum of all values ignoring all null values.
  - **VARIANCE** ([DISTINCT] | ALL] x) :
    - Returns the variance of n, ignoring all null values.

```
SQL> select avg(sal), max(sal), min(sal), sum(sal), variance(sal), stddev(sal), count(sal)
2 from emp;
```

AVG(SAL)	MAX(SAL)	MIN(SAL)	SUM(SAL)	VARIANCE(SAL)	STDDEV(SAL)	COUNT(SAL)
2032.3235	5500	99	69099	1123584.2	1059.9925	34

- The above example, shows an example of all these group functions.
- Note that the **DISTINCT** clause in all these above functions makes the function consider only non-duplicate values, while **ALL** makes it consider all values. Furthermore, the data-types for the arguments may be **CHAR**, **VARCHAR2**, **NUMBER**, or **DATE** where expr is listed.
- Also, all group functions with the exception of **COUNT(\*)** ignore null values. To substitute values in for the null values, use the **NVL** function. Recall that the **NVL** function forces group functions to include null values.

```
SQL> r
1 select avg(NVL(sal,0)), max(sal), min(sal), sum(sal), variance(sal), stddev(sal), count(*)
2* from emp
```

AVG(NVL(SAL,0))	MAX(SAL)	MIN(SAL)	SUM(SAL)	VARIANCE(SAL)	STDDEV(SAL)	COUNT(*)
1919.4167	5500	99	69099	1123584.2	1059.9925	36

- Compare the **AVG** values with the ones on top...in this case, those records having null values are treated as if their null values were '0's' and therefore, they are included in the calculation, bringing the overall average down. Note that '0' was arbitrary here...we could have substituted those null values with any other value that would have applied.
- **Creating Groups Of Data:** You can use the **GROUP BY** clause to divide the rows in a table into groups. You can then use the group functions to return summary information about each group.

```
SQL> r
1 select deptno, avg(sal), count(*)
```

```

2  from emp
3* group by deptno

```

DEPTNO	AVG (SAL)	COUNT (*)
2	1500	1
3	1500	1
10	2000	3
20	1750	4
30	1250	4
40	5250	2
	1978.8947	21

- When using the **GROUP BY** clause, make sure that all columns in the **SELECT** list that are not in the group functions are included in the **GROUP BY** clause.
- **Grouping By More Than One Column:** Sometimes there is a need to see results for groups.

```

SQL> select deptno, job, sum(sal)
2  from emp
3  group by deptno, job;

```

DEPTNO	JOB	SUM(SAL)
2	ANALYST	1500
3	ANALYST	1500
10	MANAGER	2000
10	PRESIDENT	3000
10	SALESMAN	1000
20	ANALYST	5000
20	MANAGER	2000
30	MANAGER	2000
30	SALESMAN	3000
40	ANALYST	10500
	ANALYST	34599
	SALESMAN	3000

- In the above example, the columns are organized first in order of magnitude based on the value of **DEPTNO** and secondly in alphabetical order according to the string in found in **JOB**.
- **Excluding Group Results:** In the same way that you use the **WHERE** clause to restrict the rows that you select, you used the **HAVING** clause to restrict groups. For example, to show the maximum salary of each department, but show only the departments that have a maximum salary of more than \$2900, you need to do the following.
  - Find the maximum salary for each department by grouping the department number
  - Restrict the groups to those departments with a maximum salary greater than \$2900.

```

SQL> r
1  select deptno, max(sal)

```

```

2  from emp
3  group by deptno
4* having max(sal) > 2900

```

DEPTNO	MAX(SAL)
10	3000
40	5500
	4000

- Nesting Group Functions: Group functions can be nested to a depth of two only! Here's an example...

```

SQL> select max(avg(sal))
2  from emp
3  group by deptno;

```

MAX(AVG(SAL))
5250

- The following shows an error that occurs when you try to nest at a depth greater than two...

```

SQL> r
1  select count((max(avg(sal)))
2  from emp
3* group by deptno
select count((max(avg(sal)))
                *

```

**ERROR** at line 1:

ORA-00935: **group function is nested too deeply**