Aggregate Functions, 'Group By' and 'Having' Clauses

1. AGGREGATE FUNCTIONS

In all the examples so far, we have selected values stored in each row of a table (*like SAL*), or values calculated for each row (*like SAL*12*). That is, we have selected information about *individual* rows stored in the database. You can also select "summary" information about *groups* of rows in the database.

Oracle provides five aggregate functions (*or group functions*) that can be applied to data retrieved in a query:

AVG	Computes the average value
SUM	Computes the total value
MIN	Finds the minimum value
MAX	Finds the maximum value
COUNT	Counts the number of values

1.1 Selecting Summary Information from One Group

As shown below, the column that the function is applied to must be enclosed in parentheses.

- Find the average salary for clerks

```
SQL > SELECT AVG(SAL)
      2 FROM EMP
      3 WHERE JOB = 'CLERK';
```

Oracle uses all the rows that satisfy the search-condition (*WHERE JOB = 'CLERK'*) to compute the summary information. Rather than displaying a value (*SAL*) for each individual row (*CLERK*) selected, Oracle calculates a single value – AVG(*SAL*) – as a result.

You can have more than one group function in a SELECT clause.

- Find the total salary and total commission for salesmen

```
SQL > SELECT SUM(SAL), SUM(COMM)
      2 FROM EMP
      3 WHERE JOB = 'SALESMAN';
```

You can use group functions in arithmetic expressions.

- Compute the average annual salary plus commission for all salesmen.

```
SQL > SELECT AVG(SAL + COMM)*12
      2 FROM EMP
      3 WHERE JOB = 'SALESMAN';
```

- Find the highest and lowest paid employee salaries and the difference between them.

```
SQL > SELECT MAX(SAL), MIN(SAL), MAX(SAL) – MIN(SAL)
      2 FROM EMP;
```

You can use group functions with arithmetic or string functions.

- Find the number of characters in the longest department name.

```
SQL > SELECT MAX(LENGTH(DNAME))
      2 FROM DEPT;
```

In standard SQL, you may NOT have both aggregates and non-aggregates in the same SELECT list. For example, SELECT ENAME, AVG(*SAL*) is an error. This is because ENAME is an attribute of *each* row selected and AVG(*SAL*) is an attribute of *all* the rows selected. So if you want to find the name and salary of the employee (*or employees*) who receive the maximum salary, you cannot use the above query.

Instead, you should use a sub-query as in the following example.

- Find the name and salary of the employee (*or employees*) who receive the maximum salary.

```
SQL > SELECT ENAME, JOB, SAL
      2 FROM EMP
      3 WHERE SAL =
      4 (SELECT MAX(SAL) FROM EMP);
```

The COUNT function can be used to count the number of values, number of distinct values, or number of rows selected by the query.

- Count the number of employees who receive a commission

```
SQL > SELECT COUNT (COMM)
      2 FROM EMP;
```

COUNT may be used with the keyword DISTINCT to return the number of different values stored in the column. That is, duplicate values are eliminated before values are counted.

- Count the number of different jobs held by employees in department 30

```
SQL > SELECT COUNT(DISTINCT JOB)
      2 FROM EMP
      3 WHERE DEPTNO = 30;
```

- Count the number of employees in department 30

```
SQL > SELECT COUNT (*)
      2 FROM EMP
      3 WHERE DEPTNO = 30;
```

2 SELECTING SUMMARY INFORMATION FROM GROUPS – GROUP BY

Let's say you want to know the average salary for the employees in department 10, department 20 and department 30. You could solve this problem with the following three queries:

- Find the average salary of the employees in each department.

```
SQL > SELECT AVG (SAL)
      2 FROM EMP
      3 WHERE DEPTNO = 10;
SQL > SELECT AVG (SAL)
      2 FROM EMP
      3 WHERE DEPTNO = 20;
```

```
SQL > SELECT AVG (SAL)
    2 FROM EMP
    3 WHERE DEPTNO = 30;
```

The same information could be returned by a single query with a GROUP BY clause. The GROUP BY clause divides a table into groups of rows with matching values in the same column (*or columns*).

- List the department number and average salary of each department

```
SQL > SELECT DEPTNO, AVG(SAL)
    2 FROM EMP
    3 GROUP BY DEPTNO;
```

In the above query, all the employees are divided into groups based on their department number (*GROUP BY DEPTNO*). Thus, there are three different groups: the employees in department 10, the employees in department 20 and the employees in department 30. The group functions AVG(*SAL*) is then applied to all the rows in each group.

When you have a GROUP BY clause you can select the group column in addition to group functions. This is because the grouping column is an attribute of the group (*all the rows of the group have the same value for the grouping column*).

The GROUP BY clause always follows the WHERE clause, or the FROM clause when there is no WHERE clause in the command.

- Find each department's average annual salary for all its employees except the managers and the president

```
SQL > SELECT DEPTNO, AVG(SAL)*12
    2 FROM EMP
    3 WHERE JOB NOT IN ('MANAGER', 'PRESIDENT')
    4 GROUP BY DEPTNO;
```

You can partition rows of a table into groups based on the values in more than one column.

- Divide all employees into groups by department, and by jobs within department. Count the employees in each group and compute each group's average annual salary.

```
SQL > SELECT DEPTNO, JOB, COUNT(*), AVG(SAL)*12
  2 FROM EMP
  3 GROUP BY DEPTNO, JOB;
```

In the example above, a group is formed for every job within every department (*GROUP BY DEPTNO, JOB*). Look at the last row of the query result. This group of department 30 salesmen has a count of 4 and an average salary of $16,800.

You can have join-conditions and group functions in the same query.

- Issue the same query as above except list the department name rather than the department number

```
SQL > SELECT DNAME, JOB, COUNT(*), AVG(SAL)*12
  2 FROM EMP, DEPT
  3 WHERE DEPT.DEPTNO = EMP.DEPTNO
  4 GROUP BY DNAME, JOB;
```

3 SPECIFYING A SEARCH-CONDITION FOR GROUPS – HAVING

You can specify search-conditions for groups just as you can specify conditions for individual rows. As you know, search-conditions for individual rows are specified in the WHERE clause, for groups, you should use HAVING clause to do specification.

- List the average annual salary for all job groups having more than 2 employees in the group.

```
SQL > SELECT JOB, COUNT(*), AVG(SAL)*12
  2 FROM EMP
  3 GROUP BY JOB
  4 HAVING COUNT(*) > 2;
```

A HAVING clause, when used, always follows the GROUP BY clause. The HAVING clause compares some property of the group (*COUNT(*) in the example above*) with a constant value (*2 in the example above*). If a group satisfies the search condition in the HAVING clause, it is included in the query result.

A query can contain both a WHERE and a HAVING clause. Firstly, the WHERE clause is applied to qualify rows; secondly, the groups are formed and the group functions are applied; thirdly, the HAVING clause is applied to qualify groups.

- List all the departments that have at least two clerks.

```
SQL > SELECT DEPTNO
      2 FROM EMP
      3 WHERE JOB = 'CLERK'
      4 GROUP BY DEPTNO
      5 HAVING COUNT(*) >=2;
```

A HAVING clause can also compare one attribute of the group with another attribute of the group.

- Find all departments with an average commission greater than 25% of average salary.

```
SQL > SELECT DEPTNO, AVG(SAL), AVG(COMM), AVG(SAL)*0.25
      2 FROM EMP
      3 GROUP BY DEPTNO
      4 HAVING AVG(COMM) > AVG(SAL)*0.25;
```

You can use a sub-query in a HAVING clause to compare an attribute of the group with a computed attribute of another group.

- List the job groups that have an average salary greater than the average salary of managers.

```
SQL > SELECT JOB, AVG(SAL)
      2 FROM EMP
      3 GROUP BY JOB
      4 HAVING AVG(SAL) >
      5 (SELECT AVG(SAL) FROM EMP
      6 WHERE JOB = 'MANAGER');
```

4 EFFECTS OF NULL VALUES ON GROUP FUNCTIONS

Null values do not participate in the computation of group functions.

- Count the number of people in department 30 who receive a salary and the number of people who receive a commission.

```
SQL > SELECT COUNT (SAL), COUNT(COMM)
  2 FROM EMP
  3 WHERE DEPTNO = 30;
```

The count of people who receive a salary, 6, is greater than the count of people who receive a commission, 3. This is because null commissions were not counted.

- Null values do not participate in the computation of any group functions.

```
SQL > SELECT SUM(SAL), COUNT(SAL), AVG(SAL), SUM(COMM),
  2 COUNT (COMM), AVG(COMM)
  3 FROM EMP
  4 WHERE DEPTNO = 30;
```

- List the average commission of employees who receive a commission, and the average commission of all employees (*treating employees who do not receive a commission as receiving a zero commission*).

```
SQL > SELECT AVG(COMM), AVG(NVL(COMM, 0))
  2 FROM EMP
  3 WHERE DEPTNO = 30;
```

5 FORMATING QUERY RESULTS INTO A REPORT

In this section, you will learn some more SQL*Plus commands to improve the appearance (*format*) of a query result. The commands to be covered in this section are:

COLUMN	Format a column's heading and data
TTITLE	Put a title on the top of the page
BTITLE	Put a title on the bottom of the page
BREAK	Break the report up into groups of rows
COMPUTE	Perform various computation in each group

5.1 Column Formatting Commands

Through the SQL*Plus COLUMN command you can change the column headings and reformat the column data in your query results.

- List the department number, name and salary of all employees in department 20.

```
SQL > SELECT DEPTNO, ENAME, SAL
      2 FROM EMP
      3 WHERE DEPTNO = 20;
```

5.1.1 Changing Column Headings

We can make the output from the previous query more readable by changing the column headings. You do this by using the COLUMN command with a HEADING clause.

Enter the command:

```
SQL > COLUMN DEPTNO HEADING DEPARTMENT
SQL > RUN
```

A column heading may be more than one word long, but if it contains a blank character then it must be enclosed in single quotes (*for example, 'EMPLOYEE NAME'*). A column heading may be displayed on more than one line. This is done by placing a *vertical bar* (|) character between the words that you want to separate lines.

```
SQL > COLUMN ENAME HEADING 'EMPLOYEE | NAME'
SQL > RUN
```

5.1.2 Reformatting Column Data

The FORMAT clause in COLUMN command allows us to change the default format of numbers and text.

- Put a heading of MONTHLY SALARY on the SAL column and format the data with a dollar sign, a comma in the thousands position, and a decimal point between the dollars and cents position.

```
SQL > COLUMN SAL HEADING 'MONTHLY|SALARY' FORMAT $99,999.99
SQL > RUN
```

The following is a list of the different format masks that can be used to control the appearance of column data:

FORMAT	VALUE	DISPLAYED
999.99	56.478	56.48
999V99	56.478	5648
9,999	8410	8,410
99999	607	607
09999	607	00607
9999	-5609	-5609
9999MI	-5629	5609-
9999PR	-5609	<5609>
B999	564	564
99.99	124.98	###.##
$99.99	45.23	$45.23
A20	Customer	Customer
A5	Customer	Custo

5.1.3 Clearing Column Attributes

To reset the display attributes for all columns (*or a specific column*), use the CLEAR command (*or CLEAR clause of the COLUMN command*).

SQL > COLUMN column_name CLEAR

or

SQL > CLEAR COLUMNS

5.2 Getting Your Report to Print on Separate Pages

- Set the 'pagesize' to 48 lines and cause a form feed to a 'newpage' prior to printing every heading.

  SQL > SET PAGESIZE 48

  SQL > SET NEWPAGE 0

  SQL > RUN

- Set the pagesize back to 24 and newpage back 1.

  SQL > SET PAGESIZE 24

  SQL > SET NEWPAGE 1

  SQL > RUN

5. 3 Page Formatting Commands

In addition to using the COLUMN command to control the appearance of individual columns on the page, there are other SQL*Plus commands that allow you to control the appearance of the page itself.

5.3.1 TTITLE

- Put a title of ACME WIDGET – PERSONAL REPORT on separate lines at the top of each page of the report.

  SQL > TTITLE 'ACME WIDGET || PERSONEL REPORT'

  SQL > SELECT DEPTNO, ENAME, SAL

  2 FROM EMP

  3 ORDER BY DEPTNO;

Notice that the bar character can be used to separate TTITLE into separated lines just as in column headings. The tow bar characters (||) caused the title to contain a blank line between "ACME WIDGET" and "PERSONAL REPORT".

5.3.2 BTITLE

- Put a title of COMPANY CONFIDENTIAL at the bottom of the page.

  SQL > BTITLE 'COMPANY CONFIDENTIAL'

  SQL > RUN

5.3.3 BREAK

The rows of a report may be 'broken-up' into groups by using the BREAK command.

BREAK ON column_name [SKIP n]

[PAGE]

- BREAK ON DEPTNO to separate the rows of the query result into groups. Skip 2 lines between each group.

  SQL > BREAK ON DEPTNO SKIP 2

  SQL > RUN

The BREAK command in the above example grouped together all the rows with the same value of DEPTNO, and only displayed it once. When a new value of DEPTNO was found, 2 lines were skipped. It is important to remember that the column that you BREAK ON should be the same as the column that you ORDER BY (otherwise the rows of your report will be broken into meaningless groups).

5.3.4 COMPUTE

After the rows of the query result have been divided into groups you can do computations for columns. Note that the COMPUTE command has no effect without a corresponding BREAK command.

- Compute the total salary of employees for each department.

  SQL > COMPUTE SUM OF SAL ON DEPTNO

  SQL > RUN

Note that subtotals were computed every time the value of DEPTNO changed.

- Compute the average salary of all employees at the end of the report.

  SQL > BREAK ON DEPTNO SKIP 2 ON REPORT

  SQL > COMPUTE AVG OF SAL ON REPORT SQL > RUN

5.4 SQL*Plus Command Buffer

Oracle saves SQL*Plus commands like BREAK and COMPUTE in several SQL*Plus command buffers (*one buffer for each different SQL*Plus command*). You can display the contents of a SQL*Plus command buffer by typing the command name by itself.

- Display the current TTITLE

  SQL > TTITLE

Oracle 'remembers' each of these commands and reuses them unless you turn them OFF or use a CLEAR command.

- Clear the current BREAK and COMPUTE commands.

  SQL > CLEAR BREAK

  SQL > CLEAR COMPUTE

  SQL > RUN

The BREAK and COMPUTE are now gone but the TTITLE and BTITLE are still on.

- Turn off TTITLE and BTITLE

  SQL > TTITLE OFF

  SQL > BTITLE OFF

  SQL > RUN