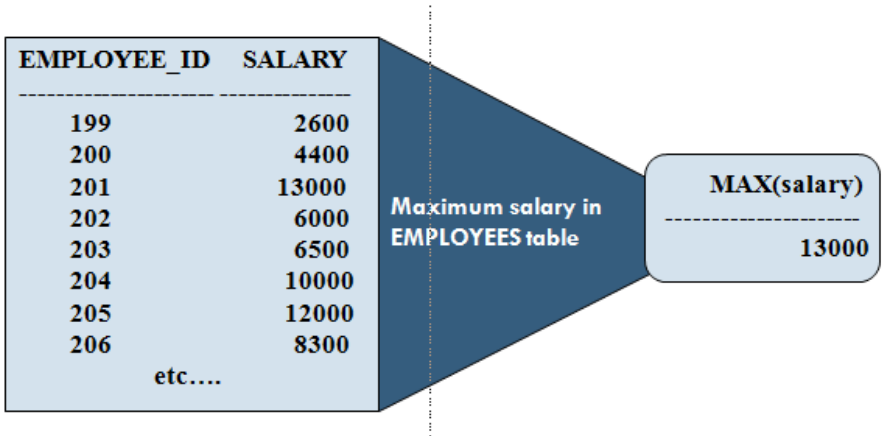
Chapter 4 (cont)
Aggregate or Group Functions

Objectives

- Identify group functions
- Describe the use of group functions
- Group data using the GROUP BY clause
- Limit grouped rows using the HAVING clause

What are Group Functions?

- Group functions operate on sets of rows to give one result per group.



Aggregate or Group Functions

- These functions operate on a collection of values and return a single value

- The five most commonly used aggregate functions are:
  - avg(): Average value
  - min(): minimum value
  - max(): maximum value
  - sum(): sum of values
  - count(): number of values
- Some examples:
  - Find the average salary for all employees whose last name begins with a 'T':

```
SELECT AVG(salary)
FROM employees
WHERE last_name LIKE 'T%';
```

– Find the number of tuples in the employees table:

SELECT COUNT(*)
FROM employees;

– Find the number of managers in the human resource database

SELECT COUNT(DISTINCT manager_id)
FROM employees;

– Find the maximum salary and the minimum salary of all employees.

SELECT MAX(salary) Maximum, MIN(salary) Minimum
FROM employees;

– If the company was to fire all employees whose last name begins with 'T', how much would be saved?

SELECT SUM(salary)
FROM employees
WHERE UPPER(last_name) LIKE 'T%';

Grouping Rows

– Creating groups of data
– Allows you to divide the table into smaller groups.
– Use the GROUP BY clause.

Aggregation – Group By

– But rather than firing all employees whose last_name begins with a 'T', it would be better to get rid of the most expensive manager and his/her staff. How to do this?
– Need a way of grouping the data, and doing calculations based on that grouping.
  – The GROUP BY clause does the job.
– Syntax is:

GROUP BY columnname [, columnname,…]

```
SELECT manager_id, SUM(salary)
FROM employees
GROUP BY manager_id
ORDER BY SUM(salary);
```

– Should note the following:
  – If a group function is used in the SELECT clause, then any individual columns listed in the SELECT clause must also be listed in the GROUP BY clause.
  – Columns used to group data in the GROUP BY clause do not have to be listed in the SELECT clause. They are only included in the SELECT clause to have the groups identified in the output.
  – Using a WHERE clause, you can exclude rows before dividing them into groups
  – Column aliases cannot be used in the GROUP BY clause.

- Another example:
  – How many employees does each manager manage?

```
SELECT manager_id, COUNT(DISTINCT(employee_id))
FROM employees
GROUP BY manager_id;
```

– **Grouping by More than One Column**
  – Sometimes need to see results for groups within groups.
  – Can return summary results for groups and subgroups by listing multiple columns in the GROUP BY.
  – Cannot guarantee the order of the result set. Should use an ORDER BY to do that.

  – Want to see the total salary for each job type within a department.

```
SELECT      department_id, job_id,
            sum(salary)
FROM        employees
GROUP BY  department_id, job_id
ORDER BY  job_id;;
```

Aggregation – Having Clause

- Is used to restrict the groups returned by a query.
- Group functions can **NOT** be used in a WHERE clause.
- Cannot use the WHERE clause to restrict groups.
- The HAVING clause serves as the WHERE clause for groups.
  - Note: can have a HAVING and a WHERE clause in the same query.
- When you use the HAVING clause, the Oracle server restricts the groups as follows:
  1. Rows are grouped.
  2. The group function is applied.
  3. Groups matching the HAVING clause are displayed.

- Example
  - Management suspects that some departments are underpaying their staff and pocketing the extra. According to their calculations, no department should have an average salary of less than $5000/month. Are there any departments that are underpaying?

```
SELECT department_id, ROUND(AVG(salary))
FROM employees
GROUP BY department_id
HAVING ROUND(AVG(salary)) < 5000;
```

- Your boss suddenly remembers that Department 10 has a lot of volunteer staff, so the salaries given are not accurate. You need to remove Department 10 from your query.

```
SELECT department_id, ROUND(AVG(salary))
FROM employees
WHERE department_id != 10
GROUP BY department_id
HAVING ROUND(AVG(salary)) < 5000;
```

Nesting Functions

- Can nest group functions just as can nest single-row functions.
- When nesting, the inner function is always resolved first.
- Single-row functions have no restrictions on the number of nesting levels.
- Group functions can only be nested to a depth of two.
- Can nest a group function in a single-row function and vice versa.

A Nesting Example

```
SELECT department_id, TO_CHAR(ROUND(AVG(NVL(salary,0))), '$99,999')
   "Average"
FROM employees
GROUP BY department_id
HAVING ROUND(AVG(salary)) < 5000;
```

Aggregate Function - Notes

- Lots of functions to choose from, including statistical etc etc.
- The AVG, SUM and statistical functions among other are only used with numeric fields
- The COUNT, MAX, and MIN functions can be applied to any data type.
  - E.g. SELECT MIN(last_name) FROM employees;
- The AVG,SUM, MAX, MIN etc functions all ignore NULL values

- COUNT(*) counts all records including those containing NULL values
- By default, AVG, SUM, MIN, MAX, COUNT etc include duplicate values. To include only unique values, need to use the DISTINCT keyword

SUMMARY

- Examined some representative group functions
- Used group functions in SELECT statements
- Grouped data using the GROUP BY clause
- Limited grouped rows using the HAVING clause