**PROGRAMMING 9**

**Database Programming with SQL  
9-1: Using GROUP BY and HAVING Clauses  
Practice Activities**

Objectives  
• Construct and execute a SQL query using GROUP BY  
• Construct and execute a SQL query using GROUP BY ... HAVING  
• Construct and execute a GROUP BY on more than one column  
• Nest group functions

Vocabulary  
Identify the vocabulary word for each definition below.

* Used to specify which groups are to be displayed; restricts groups that do not meet group criteria  
  HAVING
* Divides the rows in a table into group

GROUP BY

Try It / Solve It  
1. In the SQL query shown below, which of the following is true about this query?  
\_\_\_\_T\_\_\_ a. Kimberly Grant would not appear in the results set.  
\_\_\_\_F\_\_\_ b. The GROUP BY clause has an error because the manager\_id is not listed in the  
SELECT clause.  
\_\_\_\_T\_\_\_ c. Only salaries greater than 16001 will be in the result set.  
\_\_\_\_F\_\_\_ d. Names beginning with Ki will appear after names beginning with Ko.  
\_\_\_\_F\_\_\_ e. Last names such as King and Kochhar will be returned even if they don’t have  
salaries > 16000.  
SELECT last\_name, MAX(salary)  
FROM employees  
WHERE last\_name LIKE 'K%'  
GROUP BY manager\_id, last\_name  
HAVING MAX(salary) >16000  
ORDER BY last\_name DESC ;

2. Each of the following SQL queries has an error. Find the error and correct it. Use Oracle  
Application Express to verify that your corrections produce the desired results.

a. SELECT manager\_id  
FROM employees  
WHERE AVG(salary) <16000  
GROUP BY manager\_id;

b. SELECT cd\_number, COUNT(title)  
FROM d\_cds  
WHERE cd\_number < 93;

c. SELECT ID, MAX(ID), artist AS Artist  
FROM d\_songs  
WHERE duration IN('3 min', '6 min', '10 min')  
HAVING ID < 50  
GROUP by ID;

d. SELECT loc\_type, rental\_fee AS Fee  
FROM d\_venues  
WHERE id <100  
GROUP BY "Fee"  
ORDER BY 2;

3. Rewrite the following query to accomplish the same result:  
SELECT DISTINCT MAX(song\_id)  
FROM d\_track\_listings  
WHERE track IN ( 1, 2, 3);

SELECT track, MAX(song\_id)

FROM d\_track\_listings

WHERE track IN ( 1, 2, 3)

GROUP BY track;

4. Indicate True or False  
\_\_\_t\_\_ a. If you include a group function and any other individual columns in a SELECT clause, then each individual column must also appear in the GROUP BY clause.  
\_\_\_f\_\_ b. You can use a column alias in the GROUP BY clause.  
\_\_\_f\_\_ c. The GROUP BY clause always includes a group function.

5. Write a query that will return both the maximum and minimum average salary grouped by  
department from the employees table.

SELECT MAX(AVG(salary)) AS max\_avg\_salary,MIN(AVG(salary)) AS min\_avg\_salary

FROM employees

GROUP BY department\_id;

6. Write a query that will return the average of the maximum salaries in each department for the employees table.

SELECT AVG(MAX(salary))

FROM employees

GROUP BY department\_id;

**Database Programming with SQL  
9-2: Using ROLLUP and CUBE Operations and GROUPING SETS  
Practice Activities**

Objectives  
• Use ROLLUP to produce subtotal values  
• Use CUBE to produce cross-tabulation values  
• Use GROUPING SETS to produce a single result set  
• Use the GROUPING function to identify the extra row values created by either a ROLLUP or CUBE operation

Vocabulary  
Identify the vocabulary word for each definition below.

* Used to create subtotals that roll up from the most detailed level  
  to a grand total, following a grouping list specified in the clause  
  rollup
* An extension to the GROUP BY clause like ROLLUP that  
  produces cross-tabulation reports  
  cube
* Used to specify multiple groupings of data  
  group sets

Try It / Solve It  
1. Within the Employees table, each manager\_id is the manager of one or more employees who each have a job\_id and earn a salary. For each manager, what is the total salary earned by all of the employees within each job\_id? Write a query to display the Manager\_id, job\_id, and total salary. Include in the result the subtotal salary for each manager and a grand total of all salaries.

SELECT manager\_id,job\_id, SUM(salary) AS total\_salary

FROM employees

GROUP BY ROLLUP(manager\_id, job\_id);

2. Amend the previous query to also include a subtotal salary for each job\_id regardless of the manager\_id.

SELECT manager\_id,job\_id,SUM(salary) AS total\_salary

FROM employees

GROUP BY CUBE(manager\_id, job\_id);

3. Using GROUPING SETS, write a query to show the following groupings:  
• department\_id, manager\_id, job\_id  
• manager\_id, job\_id  
• department\_id, manager\_id

SELECT department\_id,manager\_id, job\_id, SUM(salary) AS total\_salary

FROM employees

GROUP BY GROUPING SETS (

(department\_id, manager\_id, job\_id),

(manager\_id, job\_id),

(department\_id, manager\_id)

);

**Database Programming with SQL  
9-3: Set Operators  
Practice Activities**Objectives  
• Define and explain the purpose of SET operators  
• Use a set operator to combine multiple queries into a single query  
• Control the order of rows returned using set operators

Vocabulary  
Identify the vocabulary word for each definition below.

* operator that returns all rows from both tables and eliminates  
  duplicates  
  UNION
* columns that were made up to match queries in another table  
  that are not in both tables  
  TO\_CHAR(NULL)
* operator that returns all rows from both tables, including  
  duplicates  
  UNION ALL
* used to combine results into one single result from multiple  
  SELECT statements  
  set operators
* operator that returns rows that are unique to each table  
  MINUS
* operator that returns rows common to both tables

INTERSECT

Try It / Solve It  
1. Name the different Set operators?  
union, union all, minus, intersect

2. Write one query to return the employee\_id, job\_id, hire\_date, and department\_id of all employees and a second query listing employee\_id, job\_id, start\_date, and department\_id from the job\_history table and combine the results as one single output. Make sure you suppress duplicates in the output.

SELECT employee\_id, job\_id, hire\_date

FROM employees

UNION

SELECT employee\_id, job\_id, start\_date, department\_id

FROM job\_history;

3. Amend the previous statement to not suppress duplicates and examine the output. How many extra rows did you get returned and which were they? Sort the output by employee\_id to make it easier to spot.

SELECT employee\_id, job\_id, hire\_date

FROM employees

UNION ALL

SELECT employee\_id, job\_id, start\_date, department\_id

FROM job\_history

ORDER BY employee\_id;

4. List all employees who have not changed jobs even once. (Such employees are not found in the job\_history table)

SELECT DISTINCT employee\_id

FROM employees

MINUS

SELECT DISTINCT employee\_id

FROM job\_history;

5. List the employees that HAVE changed their jobs at least once.

SELECT DISTINCT employee\_id

FROM employees

INTERSECT

SELECT DISTINCT employee\_id

FROM job\_history;

6. Using the UNION operator, write a query that displays the employee\_id, job\_id, and salary of ALL present and past employees. If a salary is not found, then just display a 0 (zero) in its place.

SELECT employee\_id, job\_id, NVL(salary, 0)

FROM employees

UNION

SELECT employee\_id, job\_id, 0

FROM job\_history

ORDER BY employee\_id;