

Chapter 11-1: Ensuring Quality Query Results

NAME: P. SUMALAVASAVI

REG NO: 192372149

Query 1:

Problem:

- Create a list of all tables whose first two characters in the name of the table is JO
- The tables must be owned by the current Oracle User

SQL Query:

```
SELECT table_name
FROM user_tables
WHERE table_name LIKE 'JO%'
ORDER BY table_name;
```

=====

Query 2:

Problem:

- Create a list that includes the first initial of every employee's first name, a space, and the last name of the employee

SQL Query:

```
SELECT SUBSTR(first_name, 1, 1) || ' ' || last_name AS "Employee_Name"
FROM employees;
```

=====

Query 3:

Problem:

- Create a list of every employee's first name concatenated to a space and the employee's last name, and the email of all employees where the email address contains the string 'IN'

SQL Query:

```
SELECT first_name || ' ' || last_name AS "Employee_Name", email AS "Email"
FROM employees
```

WHERE email LIKE '%IN%';

Query 4:

Problem:

– Create a list of 'smallest' last name and the 'highest' last name from the employees table

SQL Query:

```
SELECT  
  
    MIN(last_name) AS smallest_last_name,  
  
    MAX(last_name) AS highest_last_name  
  
FROM Employees;
```

Query 5:

Problem:

– Create a list of weekly salaries from the employees table where the weekly salary is between 700 and 3000

– The salaries should be formatted to include a \$- sign and have two decimal points like: \$9999.99

SQL Query:

```
SELECT '$' || ROUND((salary*12)/ 52, 2) AS “Weekly Salary”  
  
FROM employees  
  
WHERE (salary*12) /52 BETWEEN 700 AND 3000;
```

Query 6:

Problem:

– Create a list of every employee and his related job title sorted by job_title

SQL Query:

```
SELECT first_name || ' ' || last_name AS employee_name, job_title  
  
FROM employees
```

ORDER BY job_title;

Query 7:

Problem:

- Create a list of every employee's job, the salary ranges within the job, and the employee's salary
- List the lowest and highest salary range within each job with a dash to separate the salaries like this: 100 – 200

SQL Query:

```
SELECT SUBSTR(first_name,1,1)|| ' ' ||last_name AS "Employee Name", job_title AS  
"Job",min_salary|| '-' ||max_salary AS "salary range",salary AS "Employees salary"  
FROM employees e,jobs j  
WHERE e.job_id = j.job_id  
ORDER BY j.job_title, e.salary;
```

Query 8:

Problem:

- Using an ANSI join method, create a list of every employee's first initial and last name, and department name
- Make sure the tables are joined on all of the foreign keys declared between the two tables

SQL Query:

```
SELECT SUBSTR(e.first_name, 1, 1) || ' ' || e.last_name AS employee_name,  
d.department_name  
FROM employees e  
JOIN departments d ON e.department_id = d.department_id  
ORDER BY employee_name;
```

Query 9:

Problem:

- Change the previous listing to join only on the department_id column

SQL Query:

```
SELECT SUBSTR(e.first_name, 1, 1) || ' ' || e.last_name AS employee_name,  
d.department_name  
FROM employees e  
JOIN departments d ON e.department_id = d.department_id  
ORDER BY employee_name;
```

Query 10:

Problem:

- Create a list of every employee's last name, and the word nobody or somebody depending on whether or not the employee has a manager
- Use the Oracle DECODE function to create the list

SQL Query:

```
SELECT DECODE(manager_id, NULL, 'Nobody', 'Somebody') AS "Works for",  
last_name AS "Last Name"  
FROM employees;
```

Query 11:

Problem:

- Create a list of every employee's first initial and last name, salary, and a yes or no to show whether or not an employee makes a commission
- Fix this query to produce the result

SQL Query:

```
SELECT  
    SUBSTRING(first_name, 1, 1) AS first_initial,  
    last_name,  
    salary,  
    CASE  
        WHEN commission IS NOT NULL AND commission > 0 THEN 'yes'  
        ELSE 'no'
```

END AS makes_commission

FROM employees;

=====

Query 12:

Problem:

- Create a list of every employee's last name, department name, city, and state_province
- Include departments without employees
- An outer join is required

SQL Query:

```
SELECT e.last_name, d.department_name, d.city, d.state_province
FROM departments d
LEFT JOIN employees ON d.department_id = e.department_id;
```

=====

Query 13:

Problem:

- Create a list of every employee's first and last names, and the first occurrence of: commission_pct, manager_id, or -1
- If an employee gets commission, display the commission_pct column; if no commission, then display his manager_id; if he has neither commission nor manager, then the number -1

SQL Query:

```
SELECT
    first_name,
    last_name,
    COALESCE( CAST(commission_pct AS VARCHAR), CAST(manager_id AS
VARCHAR), '-1' ) AS first_occurrence
FROM employees;
```

=====

Query 14:

Problem:

- Create a list of every employee's last name, salary, and job_grade for all employees working in departments with a department_id greater than 50

SQL Query:

```
SELECT last_name, salary, job_grade
FROM employees
WHERE department_id > 50;
```

=====

Query 15:

Problem:

- Produce a list of every employee's last name and department name
- Include both employees without departments, and departments without employees

SQL Query:

```
SELECT e.last_name, d.department_name
FROM employees e
FULL OUTER JOIN departments ON e.department_id = d.department_id;
```

=====

Query 16:

Problem:

- Create a treewalking list of every employee's last name, his manager's last name, and his position in the company
- The top level manager has position 1, this manager's subordinates position 2, their subordinates position 3, and so on
- Start the listing with employee number 100

SQL Query:

```
WITH RECURSIVE EmployeeHierarchy AS (
    SELECT e.employee_id, e.last_name AS employee_last_name, m.last_name AS
    manager_last_name, 1 AS position
```

```

FROM employees e
LEFT JOIN employees m ON e.manager_id = m.employee_id
WHERE e.employee_id = 100
UNION ALL
SELECT e.employee_id, e.last_name AS employee_last_name, m.last_name AS
manager_last_name, eh.position + 1 AS position
FROM employees e
JOIN EmployeeHierarchy eh ON e.manager_id = eh.employee_id
LEFT JOIN employees m ON e.manager_id = m.employee_id
)
SELECT employee_last_name, manager_last_name, position
FROM EmployeeHierarchy
ORDER BY position, employee_last_name;
=====

```

Query 17:

Problem:

– Produce a list of the earliest hire date, the latest hire date, and the number of employees from the employees table

SQL Query:

```

SELECT MIN(hire_date) AS earliest_hire_date, MAX(hire_date) AS latest_hire_date,
COUNT(*) AS number_of_employees
FROM employees;
=====

```

Query 18:

Problem:

– Create a list of department names and the departmental costs (salaries added up)
– Include only departments whose salary costs are between 15000 and 31000, and sort the listing by the cost

SQL Query:

```

SELECT d.department_name, SUM(e.salary) AS departmental_cost

```

```
FROM departments d
JOIN employees e ON d.department_id = e.department_id
GROUP BY d.department_name
HAVING SUM(e.salary) BETWEEN 15000 AND 31000
ORDER BY departmental_cost;
```

Query 19:

Problem:

– Create a list of department names, the manager id, manager name (employee last name) of that department, and the average salary in each department

SQL Query:

```
SELECT d.department_name, d.manager_id, m.last_name AS manager_name,
AVG(e.salary) AS average_salary
FROM departments d
JOIN employees e ON d.department_id = e.department_id
JOIN employees m ON d.manager_id = m.employee_id
GROUP BY d.department_name, d.manager_id, m.last_name;
```

Query 20:

Problem:

– Show the highest average salary for the departments in the employees table
– Round the result to the nearest whole number

SQL Query:

```
SELECT ROUND(MAX(avg_salary)) AS highest_avg_salary
FROM (SELECT department_id, AVG(salary) AS avg_salary
FROM employees GROUP BY department_id) AS department_avg_salaries;
```

Query 21:

Problem:

- Create a list of department names and their monthly costs (salaries added up)

SQL Query:

```
SELECT d.department_name, SUM(e.salary) AS monthly_cost
FROM departments d
JOIN employees e ON d.department_id = e.department_id
GROUP BY d.department_name;
```

=====

Query 22:

Problem:

- Create a list of department names, and job_ids
- Calculate the monthly salary cost for each job_id within a department, for each department, and for all departments added together

SQL Query:

```
WITH JobCosts AS (
    SELECT d.department_name, e.job_id, SUM(e.salary) AS monthly_cost
    FROM departments d
    JOIN employees e ON d.department_id = e.department_id
    GROUP BY d.department_name, e.job_id
),
DepartmentCosts AS (
    SELECT department_name, SUM(monthly_cost) AS department_cost
    FROM JobCosts
    GROUP BY department_name
),
TotalCost AS (
    SELECT SUM(monthly_cost) AS total_cost
    FROM JobCosts
)
SELECT jc.department_name, jc.job_id,
```

```

    jc.monthly_cost AS job_cost_within_department, dc.department_cost, tc.total_cost
FROM JobCosts jc
JOIN DepartmentCosts dc ON jc.department_name = dc.department_name
CROSS JOIN TotalCost tc
ORDER BY jc.department_name, jc.job_id;
=====

```

Query 23:

Problem:

- Create a list of department names, and job_ids
- Calculate the monthly salary cost for each job_id within a department, for each department, for each group of job_ids irrespective of the department, and for all departments added together (Hint: Cube)

SQL Query:

```

SELECT department_name, job_id, SUM(salary) AS monthly_cost
FROM departments d
JOIN employees e ON d.department_id = e.department_id
GROUP BY CUBE (department_name, job_id);
=====

```

Query 24:

Problem:

- Expand the previous list to also show if the department_id or job_id was used to create the subtotals shown in the output (Hint: Cube, Grouping)

SQL Query:

```

SELECT department_name, job_id, SUM(salary) AS monthly_cost,
    CASE WHEN GROUPING(department_name) = 1 THEN 'Subtotal or Total'
    ELSE 'Detail'
END AS department_summary,
    CASE WHEN GROUPING(job_id) = 1 THEN 'Subtotal or Total'
    ELSE 'Detail'
END AS job_summary

```

```
FROM departments d
JOIN employees e ON d.department_id = e.department_id
GROUP BY CUBE (department_name, job_id);
```

Query 25:

Problem:

- Create a list that includes the monthly salary costs for each job title within a department
- In the same list, display the monthly salary cost per city. (Hint: Grouping Sets)

SQL Query:

```
SELECT d.department_name, e.job_id, d.city, SUM(e.salary) AS monthly_cost
FROM departments d
JOIN employees e ON d.department_id = e.department_id
GROUP BY GROUPING SETS ((d.department_name, e.job_id), (d.city));
```

Query 26:

Problem:

- Create a list of employee names as shown and department ids
- In the same report, list the department ids and department names. And finally, list the cities
- The rows should not be joined, just listed in the same report. (Hint: Union)

SQL Query:

```
SELECT CONCAT(e.first_name, ' ', e.last_name) AS employee_name,
e.department_id AS department_id
FROM employees e
UNION ALL
SELECT CAST(d.department_id AS VARCHAR) AS department_id,
d.department_name AS department_name
FROM departments d
UNION ALL
```

```
SELECT NULL AS employee_name, NULL AS department_id, d.city AS city  
FROM departments d;
```

Query 27:

Problem:

– Create a list of each employee's first initial and last name, salary, and department name for each employee earning more than the average for his department

SQL Query:

```
SELECT SUBSTRING(e.first_name, 1, 1) AS first_initial, e.last_name, e.salary,  
d.department_name  
FROM employees e  
JOIN departments d ON e.department_id = d.department_id  
WHERE e.salary > (  
    SELECT AVG(e2.salary)  
    FROM employees e2  
    WHERE e2.department_id = e.department_id  
);
```