

Database Programming with SQL

11-1 Ensuring Quality Query Results

1. Write the Query • 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 • Tables Used: –User_tables

```
1 CREATE TABLE user_tables (  
2     table_name VARCHAR2(30)  
3 );  
4
```

2. Creating employees table

```
CREATE TABLE employees (  
    employee_id NUMBER PRIMARY KEY,  
    first_name VARCHAR2(50),  
    last_name VARCHAR2(50),  
    email VARCHAR2(100),  
    phone_number VARCHAR2(20),  
    hire_date DATE,  
    job_id VARCHAR2(10),  
    salary NUMBER(8, 2),  
    commission_pct NUMBER(2, 2),  
    manager_id NUMBER,  
    department_id NUMBER  
);
```

3. Creating jobs table

```
CREATE TABLE Jobys (  
    job_id VARCHAR2(10) PRIMARY KEY,  
    job_title VARCHAR2(35),  
    min_salary NUMBER,  
    max_salary NUMBER  
);
```

4. Creating departments table

```
CREATE TABLE departments (  
    department_id NUMBER PRIMARY KEY,  
    department_name VARCHAR2(30),  
    manager_id NUMBER,  
    location_id NUMBER  
);
```

5. Creating locations table

```
CREATE TABLE locations (  
    location_id NUMBER PRIMARY KEY,  
    street_address VARCHAR2(40),  
    postal_code VARCHAR2(12),  
    city VARCHAR2(30),  
    state_province VARCHAR2(25),  
    country_id VARCHAR2(2)  
);
```

6. Creating job_grades table

```
CREATE TABLE job_gradess (  
    grade_level VARCHAR2(3) PRIMARY KEY,  
    lowest_sal NUMBER,  
    highest_sal NUMBER  
);
```

7. Inserting sample data into the tables

```
INSERT INTO user_tables (table_name) VALUES ('JOHN_TABLE');  
INSERT INTO user_tables (table_name) VALUES ('JO_ORDERS');  
INSERT INTO user_tables (table_name) VALUES ('EMPLOYEES');  
INSERT INTO user_tables (table_name) VALUES ('JOBS');
```

```
INSERT INTO employees VALUES (100, 'Steven', 'King', 'SKING', '515.123.4567', TO_DATE('17-JUN-1987', 'DD-MON-YYYY'), 'AD_PRES', 24000, NULL, NULL, 90);
INSERT INTO employees VALUES (101, 'Neena', 'Kochhar', 'NKOCHHAR', '515.123.4568', TO_DATE('21-SEP-1989', 'DD-MON-YYYY'), 'AD_VP', 17000, NULL, 100, 90);
INSERT INTO employees VALUES (102, 'Lex', 'De Haan', 'LDEHAAN', '515.123.4569', TO_DATE('13-JAN-1993', 'DD-MON-YYYY'), 'AD_VP', 17000, NULL, 100, 90);
```

```
INSERT INTO jobys VALUES ('AD_PRES', 'President', 20000, 40000);
INSERT INTO jobys VALUES ('AD_VP', 'Administration Vice President', 15000, 30000);
INSERT INTO jobys VALUES ('IT_PROG', 'Programmer', 4000, 10000);
```

```
INSERT INTO departments VALUES (90, 'Executive', 100, 1700);
INSERT INTO departments VALUES (60, 'IT', 103, 1700);
INSERT INTO departments VALUES (50, 'Purchasing', 114, 1700);
```

```
INSERT INTO locations VALUES (1700, '2004 Charade Rd', '98199', 'Seattle', 'WA', 'US');
INSERT INTO locations VALUES (1800, '2014 Jabberwocky Rd', '26192', 'Southlake', 'TX', 'US');

INSERT INTO job_grades
VALUES ('A', 10000, 20000);

INSERT INTO job_grades
VALUES ('B', 8000, 12000);

INSERT INTO job_grades
VALUES ('C', 6000, 8000);
```

1. List of tables whose name starts with "JO"

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

2. List with the first initial of every employee's first name and the last name

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

3. List of employees' full names and emails containing 'IN'

```
SELECT first_name || ' ' || last_name AS full_name, email
FROM employees
WHERE email LIKE '%IN%';
```

4. Smallest and highest last names

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

5. List of weekly salaries between \$700 and \$3000

```
SELECT TO_CHAR(salary / 4, '$9999.99') AS weekly_salary
FROM employees
WHERE salary / 4 BETWEEN 700 AND 3000;
```

6. List of employees and their job titles sorted by job title

```
SELECT e.first_name || ' ' || e.last_name AS employee_name, j.job_title
FROM employees e
JOIN jobs j ON e.job_id = j.job_id
ORDER BY j.job_title;
```

7. List of employees' jobs, salary ranges within jobs, and the employee's salary

```
SELECT e.job_id, MIN(e.salary) || ' - ' || MAX(e.salary) AS salary_range, e.salary
FROM employees e
JOIN jobs j ON e.job_id = j.job_id
GROUP BY e.job_id, e.salary;
```

8. List of employees' first initial, last name, and department name using ANSI join

```
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;
```

9. List of employees' first initial, last name, and department name joined only on department_id

```
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;
```

10. List of employees' last names and whether they have a manager using DECODE

```
SELECT last_name, DECODE(manager_id, NULL, 'nobody', 'somebody') AS manager_status
FROM employees;
```

11. List of employees' first initial, last name, salary, and if they make commission (fixed query)

```
SELECT SUBSTR(first_name, 1, 1) || ' ' || last_name AS "Employee Name", salary AS "Salary",
       DECODE(commission_pct, NULL, 'No', 'Yes') AS 'Commission'
FROM employees;
```

12. List of employees' last name, department name, city, and state_province including departments without employees

```
SELECT e.last_name, d.department_name, l.city, l.state_province
FROM employees e
RIGHT JOIN departments d ON e.department_id = d.department_id
JOIN locations l ON d.location_id = l.location_id;
```

13. List of employees' first and last names, first occurrence of commission_pct, manager_id, or -1

```
SELECT first_name, last_name,
       COALESCE(TO_CHAR(commission_pct), TO_CHAR(manager_id), '-1') AS result
FROM employees;
```

14. List of employees' last name, salary, and job_grade for departments with department_id > 50

```
SELECT e.last_name, e.salary, jg.grade_level
FROM employees e
JOIN job_grades jg ON e.salary BETWEEN jg.lowest_sal AND jg.highest_sal
WHERE e.department_id > 50;
```

15. List of employees' last name and department name including both employees without departments and departments without employees

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

16. Treewalking list of employees' last name, their manager's last name, and their position

```
SELECT e.last_name,
       (SELECT last_name FROM employees m WHERE m.employee_id = e.manager_id) AS manager_la
       LEVEL AS position
FROM employees e
START WITH employee_id = 100
CONNECT BY PRIOR employee_id = manager_id;
```

17. Earliest hire date, latest hire date, and number of employees

```
SELECT MIN(hire_date) AS lowest_hire_date, MAX(hire_date)
AS highest_hire_date, COUNT(*) AS no_of_employees
FROM employees;
```

18. List of department names and departmental costs between \$15000 and \$31000, sorted by cost

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

19. List of department names, manager id, manager name, and average salary

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

20. Highest average salary for departments rounded to the nearest whole number

```
SELECT ROUND(MAX(AVG(e.salary)) OVER
(PARTITION BY e.department_id)) AS highest_avg_salary
FROM employees e;
```

21. List of department names and their monthly costs

```
SELECT d.department_name, SUM(e.salary / 12) AS monthly_cost
FROM employees e
JOIN departments d ON e.department_id = d.department_id
GROUP BY d.department_name;
```

22. List of department names, job_ids, and monthly salary cost for each job_id within a department, and for all departments

```
SELECT d.department_name, e.job_id, SUM(e.salary / 12) AS monthly_cost
FROM employees e
JOIN departments d ON e.department_id = d.department_id
GROUP BY ROLLUP(d.department_name, e.job_id);
```

23. Expanded list with CUBE and GROUPING

```
SELECT d.department_name, e.job_id, SUM(e.salary / 12) AS monthly_cost,
       GROUPING(d.department_name) AS dept_grouping, GROUPING(e.job_id) AS
FROM employees e
JOIN departments d ON e.department_id = d.department_id
GROUP BY CUBE(d.department_name, e.job_id);
```


24. Monthly salary costs for each job title within a department and per city

```
SELECT d.department_name, e.job_id, SUM(e.salary / 12) AS monthly_cost, l.city
FROM employees e
JOIN departments d ON e.department_id = d.department_id
JOIN locations l ON d.location_id = l.location_id
GROUP BY GROUPING SETS ((d.department_name, e.job_id), (l.city));
```

25. Employee names and department ids, department ids and department names, and cities using UNION

```
SELECT first_name || ' ' || last_name AS employee_name, department_id
FROM employees
UNION
SELECT TO_CHAR(department_id), department_name
FROM departments
UNION
SELECT city, NULL
FROM locations;
```

26. List of employees' first initial, last name, salary, and department name for those earning more than the department average



```
1 SELECT SUBSTR(e.first_name, 1, 1) || ' ' || e.last_name
2 AS employee_name, e.salary, d.department_name
3 FROM employees e
4 JOIN departments d ON e.department_id = d.department_id
5 WHERE e.salary > (SELECT AVG(salary) FROM employees
6 WHERE department_id = e.department_id);
7
```