**Chapter – 11-1- Farming Quality Query Results**

**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

**SELECT table\_name**

**FROM user\_tables**

**WHERE table\_name LIKE 'JO%'**

**ORDER BY table\_names;**

**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

**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'

**SELECT CONCAT(e.first\_name, ' ') || e.last\_name AS full\_name,**

**e.email**

**FROM employees e**

**WHERE e.email LIKE '%IN%';**

**Order BY email;**

**Query-4**

**Problem:**

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

**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

**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

**SELECT SUBSTR(first\_name,1,1)||’ ‘ || last\_name AS “Employee Name”,job\_title AS “Job”**

**FROM employees e,jobs j**

**WHERE e.job\_id = j.job\_id**

**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

**SELECT SUBSTR(first\_name,1,1)||’ ‘ || last\_name AS “Employee Name”,job\_title AS “Job”,min\_salary||’-‘||max\_salary AS “Salary Range”,salary AS “Employee Salary”**

**FROM employees e,jobs j**

**WHERE e.job\_id = j.job\_id**

**ORDER BY j.job\_title;**

**Query-8**

**Problem:**

**−** Using an ANSII 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

**SELECT SUBSTR(first\_name, 1, 1) ||’ ‘||**

**last\_name AS “Employee Name”,**

**department\_name AS “Department Name”**

**FROM employees JOIN departments ;**

**Query-9**

**Problem:**

− Change the previous listing to join only on the department\_id column

**SELECT SUBSTR(first\_name, 1, 1) ||’ ‘||**

**last\_name AS “Employee Name”,**

**department\_name AS “Department Name”**

**FROM employees JOIN departments**

**USING(department\_id);**

**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

**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

**SELECT SUBSTR(first\_name, 1, 1) ||’ ‘||**

**last\_name AS “Employee Name”,**

**salary AS “Salary”,**

**NVL2(commission\_pct, 'YES', 'NO') AS 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

**SELECT e.last\_name,**

**d.department\_name,**

**d.city,**

**d.state\_province**

**FROM departments d**

**LEFT OUTER JOIN employees e**

**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

**SELECT first\_name,**

**last\_name,**

**COALESCE(commission\_pct, manager\_id, -1) AS commission\_or\_manager**

**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

**SELECT e.last\_name,**

**e.salary,**

**j.job\_grade**

**FROM employees e**

**JOIN jobs j ON e.job\_id = j.job\_id**

**WHERE e.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

**SELECT e.last\_name,**

**d.department\_name**

**FROM employees e**

**FULL OUTER JOIN departments d**

**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

**WITH RECURSIVE employee\_hierarchy AS (-- Anchor member: start with the specific employee**

**SELECT e.employee\_id,e.last\_name, e.manager\_id,1 AS position FROM employees e**

**WHERE e.employee\_id = 100**

**UNION ALL**

**-- Recursive member: find subordinates of the current level**

**SELECT e.employee\_id,e.last\_name,e.manager\_id, eh.position + 1 AS position**

**FROM employees e**

**JOIN employee\_hierarchy eh ON e.manager\_id = eh.employee\_id**

**)**

**SELECT eh.last\_name AS employee\_last\_name, m.last\_name AS manager\_last\_name, eh.position**

**FROM employee\_hierarchy eh**

**LEFT JOIN employees m**

**ON eh.manager\_id = m.employee\_id**

**ORDER BY eh.position, eh.employee\_id;**

**Query-17**

**Problem:**

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

**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

**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 31000ORDER 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

**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**

**LEFT 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

**SELECT ROUND(MAX(avg\_salary)) AS highest\_avg\_salary**

**FROM (**

**SELECT department\_id, AVG(salary) AS avg\_salary**

**FROM employees**

**GROUP BY department\_id**

**);**

**Query-21**

**Problem:**

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

**SELECT d.department\_name,SUM(e.salary) AS monthly\_cost**

**FROM departments d**

**LEFT 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

**SELECT d.department\_name, e.job\_id,**

**SUM(e.salary) AS monthly\_salary\_cost**

**FROM departments d**

**JOIN employees e ON d.department\_id = e.department\_id**

**GROUP BY d.department\_name, e.job\_id**

**ORDER BY d.department\_name, e.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)

**SELECT d.department\_name,e.job\_id,**

**SUM(e.salary) AS monthly\_salary\_cost**

**FROM departments d**

**JOIN employees e ON d.department\_id = e.department\_id**

**GROUP BY CUBE(d.department\_name, e.job\_id)**

**ORDER BY d.department\_name, e.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)

**SELECT d.department\_name,e.job\_id,**

**SUM(e.salary) AS monthly\_salary\_cost,CASE**

**WHEN GROUPING(d.department\_name) = 1 THEN 'Subtotal'**

**WHEN GROUPING(e.job\_id) = 1 THEN 'Subtotal'**

**ELSE 'Detail'**

**END AS row\_type**

**FROM departments d**

**JOIN employees e ON d.department\_id = e.department\_id**

**GROUP BY CUBE(d.department\_name, e.job\_id)**

**ORDER BY d.department\_name, e.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)

**SELECT d.department\_name,e.job\_id, e.city,**

**SUM(e.salary) AS monthly\_salary\_cost,**

**CASE**

**WHEN GROUPING(d.department\_name) = 1 AND GROUPING(e.job\_id) = 1 AND GROUPING(e.city) = 1 THEN 'Total'**

**WHEN GROUPING(e.job\_id) = 1 AND GROUPING(e.city) = 0 THEN 'Job Title in Department'**

**WHEN GROUPING(d.department\_name) = 0 AND GROUPING(e.city) = 1 THEN 'City Total'**

**ELSE 'Detail'**

**END AS row\_type**

**FROM departments d**

**JOIN employees e ON d.department\_id = e.department\_id**

**GROUP BY**

**GROUPING SETS ((d.department\_name, e.job\_id),**

**(e.city),(d.department\_name, e.city),**

**()**

**)**

**ORDER BY d.department\_name, e.job\_id, e.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)

-- Query to list employee names and department IDs

**SELECT 'Employee' AS entity\_type,**

**e.first\_name || ' ' || e.last\_name AS name\_or\_description,**

**e.department\_id**

**FROM employees e**

**UNION ALL**

**-- Query to list department IDs and department names**

**SELECT 'Department' AS entity\_type,**

**d.department\_id AS name\_or\_description,**

**NULL AS department\_id -- This column is NULL because it's not applicable here**

**FROM departments d**

**UNION ALL**

**-- Query to list cities**

**SELECT 'City' AS entity\_type,e.city AS name\_or\_description,**

**NULL AS department\_id -- This column is NULL because it's not applicable here**

**FROM employees e**

**ORDER BY entity\_type, name\_or\_description;**

**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

**SELECT SUBSTR(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**

**)**

**ORDER BY e.last\_name, first\_initial;**