**11-1 : Ensuring 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\_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**

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 first\_name || ' ' || last\_name AS “Employee name”,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.**

SELECT MIN(last\_name),MAX(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 TO\_CHAR(salary / 52, '$9999.99') AS weekly\_salary

FROM employees

WHERE (salary / 52) BETWEEN 700 AND 3000;

(Or)

SELECT ‘$’||ROUND((salary\*12)/52,2) AS weekly\_salary

FROM employees

WHERE (salary\*12)/52 ETWEEN 700 AND 3000;

**QUERY-6:**

**Problem:**

**− Create a list of every employee and his related job title sorted by job\_title**

SELECT e.first\_name || ' ' || e.last\_name AS full\_name, j.job\_title

FROM employees e JOIN jobs j ON e.job\_id = j.job\_id

ORDER BY j.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 “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 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 (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;

**QUERY-9:**

**Problem:**

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

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;

**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 last\_name,DECODE(manager\_id, NULL, 'nobody', 'somebody') AS manager\_status

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,NVL2(commission\_pct, 'yes', 'no') AS commission\_status

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, l.city, l.state\_province

FROM departments d LEFT JOIN employees e ON d.department\_id = e.department\_id LEFT JOIN locations l ON d.location\_id = l.location\_id

ORDER BY d.department\_name, e.last\_name;

**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,

CASE

WHEN commission\_pct IS NOT NULL THEN commission\_pct

WHEN manager\_id IS NOT NULL THEN manager\_id

ELSE -1

END AS result

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 departments d LEFT JOIN employees e ON d.department\_id = e.department\_id

ORDER BY d.department\_name, e.last\_name**;**

**QUERY-16:**

**•Problem:**

**−Create a treewalkinglist 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 EmployeeHierarchy AS

(SELECT e.employee\_id, e.last\_name AS employee\_last\_name, e.manager\_id,NULL AS manager\_last\_name, 1 AS position

FROM employees e

WHERE e.employee\_id = 100

UNION ALL

SELECT e.employee\_id,e.last\_name AS employee\_last\_name,

e.manager\_id,eh.employee\_last\_name AS manager\_last\_name,

eh.position + 1

FROM employees e INNER JOIN EmployeeHierarchy eh ON e.manager\_id = eh.employee\_id)

SELECT eh.employee\_id,eh.employee\_last\_name,eh.manager\_last\_name,

eh.position

FROM EmployeeHierarchy eh

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 department\_cost

FROM employees e INNER 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 department\_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 INNER JOIN employees m ON d.manager\_id = m.employee\_id INNER JOIN employees e ON d.department\_id = e.department\_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\_average\_salary

FROM ( SELECT AVG(salary) AS avg\_salary FROM employees

GROUP BY department\_id) subquery;

**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 employees e INNER JOIN departments d ON e.department\_id = d.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\_idwithin a department, for each department, and for all departments added together**

**-- Monthly salary cost for each job\_id within a department**

WITH DepartmentJobCosts AS ( SELECTd.department\_name,e.job\_id,

SUM(e.salary / 12) AS monthly\_cost FROM employees e INNER JOIN

departments d ON e.department\_id = d.department\_id

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

-- Monthly salary cost for each department

DepartmentCosts AS ( SELECT department\_name, NULL AS job\_id,

SUM(monthly\_cost) AS monthly\_cost

FROM DepartmentJobCosts

GROUP BY department\_name

)

-- Monthly salary cost for all departments

, TotalCost AS (SELECT 'All Departments' AS department\_name, NULL AS job\_id, SUM(monthly\_cost) AS monthly\_cost FROM DepartmentCosts )

SELECT department\_name, job\_id, monthly\_cost

FROM DepartmentJobCosts UNION ALL SELECT department\_name, job\_id, monthly\_cost FROM DepartmentCosts UNION ALL SELECT department\_name, job\_id, monthly\_cost FROM TotalCost

ORDER BY department\_name, job\_id;

**QUERY-23:**

**•Problem:**

**−Create a list of department names, and job\_ids**

**−Calculate the monthly salary cost for each job\_idwithin a department, for each department, for each group of job\_idsirrespective of the department, and for all departments added together (Hint: Cube)**

SELECT department\_name, job\_id,SUM(salary / 12) AS monthly\_cost

FROM employees e INNER JOIN departments d ON e.department\_id = d.department\_id

GROUP BY CUBE(department\_name, job\_id)

ORDER BY department\_name, job\_id;

**QUERY-24:**

**Problem: − Expand the previous list to also show if the department\_idor job\_id was used to create the subtotals shown in the output (Hint: Cube, Grouping)**

SELECT department\_name, job\_id, SUM(salary / 12) AS monthly\_cost,

GROUPING(department\_name) AS is\_department\_total,

GROUPING(job\_id) AS is\_job\_total

FROM employees e INNER JOIN departments d ON e.department\_id = d.department\_id

GROUP BY CUBE(department\_name, job\_id)

ORDER BY 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)**

SELECT department\_name, job\_title, city, SUM(salary / 12) AS monthly\_cost

FROM employees e INNER JOIN departments d ON e.department\_id = d.department\_id

INNER JOIN locations l ON d.location\_id = l.location\_id

GROUP BY

GROUPING SETS ( (department\_name, job\_title), (city))

ORDER BY

department\_name, job\_title, 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)**

**-- List of employee names and department IDs**

SELECT e.employee\_name AS name, e.department\_id AS id,NULL AS department\_name,NULL AS city

FROM employees e

UNION ALL

-- List of department IDs and department names

SELECT NULL AS name,

d.department\_id AS id, d.department\_name AS department\_name,

NULL AS city

FROM departments d

UNION ALL

SELECT NULL AS name, NULL AS id, NULL AS department\_name, l.city AS city

FROM locations l

ORDER BY name, id, department\_name, city;

**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(salary) FROM EmployeesWHERE department\_id = e.department\_id);