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

SQL Query

SELECT table\_name

FROM user\_tables

WHERE table\_name LIKE 'JO%';

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 **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** TO\_CHAR(weekly\_salary, '$9999.99') **AS** "Formatted Weekly Salary"

**FROM** Employees

**WHERE** weekly\_salary 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** e.first\_name || ' ' || e.last\_name **AS** "Employee Name", j.job\_title **AS** "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

SQL Query

**SELECT** j.job\_title **AS** "Job Title", j.min\_salary || ' – ' || j.max\_salary **AS** "Salary Range", e.salary **AS** "Employee Salary"

**FROM** Employees e **JOIN** Jobs j **ON** 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

SQL Query

**SELECT** SUBSTR(e.first\_name, 1, 1) || ' ' || e.last\_name **AS** "Employee Name",d.department\_name **AS** "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

SQL Query

**SELECT** SUBSTR(e.first\_name, 1, 1) || ' ' || e.last\_name **AS** "Employee Name", d.department\_name **AS** "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

SQL Query

**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

SQL Query

**SELECT** SUBSTR(first\_name, 1, 1) || ' ' || last\_name **AS** "Employee Name", salary, DECODE(commission\_pct, NULL, 'no', 'yes') **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

SQL Query

**SELECT** e.last\_name **AS** "Last Name", d.department\_name **AS** "Department Name", l.city **AS** "City", l.state\_province **AS** "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;

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**(commission\_pct, manager\_id, -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** e.last\_name **AS** "Last Name", e.salary **AS** "Salary", j.job\_grade **AS** "Job Grade"

**FROM** Employees e **JOIN** job\_grades j **ON** e.job\_grade = j.job\_grade

**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

SQL Query

**SELECT** e.last\_name **AS** "Last Name", d.department\_name **AS** "Department Name"

**FROM** Employees e

**FULL** **OUTER** **JOIN** Departments d **ON** e.department\_id = d.department\_id

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

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 (

-- Anchor member: Start with employee number 100

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

(CASE WHEN e.employee\_id = 100 THEN 1 ELSE 0 END) AS position

FROM Employees e

WHERE e.employee\_id = 100

UNION ALL-- Recursive member: Get employees under the current employee

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

FROM Employees e

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

SELECT eh.employee\_last\_name AS "Employee Last Name",

(SELECT last\_name FROM Employees WHERE employee\_id = eh.manager\_id) AS "Manager Last Name",eh.position AS "Position"

FROM EmployeeHierarchy eh

ORDER BY eh.position, eh.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 AS "Department Name",SUM(e.salary) AS "Department 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 "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

SQL Query

SELECT d.department\_name AS "Department Name",d.manager\_id AS "Manager ID",e.last\_name AS "Manager Name",AVG(e2.salary) AS "Average Salary"

FROM Departments d

LEFT JOIN Employees e ON d.manager\_id = e.employee\_id

LEFT JOIN Employees e2 ON d.department\_id = e2.department\_id

GROUP BY d.department\_name, d.manager\_id, e.last\_name

ORDER BY d.department\_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 Average 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)

SQL Query

SELECT d.department\_name AS "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

ORDER 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 MonthlyCosts AS (SELECT d.department\_name AS "Department Name",e.job\_id AS "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)

SELECT "Department Name","Job ID","Monthly Salary Cost"

FROM MonthlyCosts

UNION ALL

SELECT 'Total' AS "Department Name",NULL AS "Job ID",

SUM("Monthly Salary Cost") AS "Monthly Salary Cost"

FROM MonthlyCosts

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\_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 d.department\_name AS "Department Name",e.job\_id AS "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)

SQL Query

SELECT d.department\_name AS "Department Name",e.job\_id AS "Job ID",SUM(e.salary) AS "Monthly Salary Cost",CASE WHEN GROUPING(d.department\_name) = 1 THEN 'Total for Job ID'WHEN GROUPING(e.job\_id) = 1 THEN 'Total for Department'

ELSE 'Detail'

END AS "Subtotal 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)

SQL Query

SELECT d.department\_name AS "Department Name",e.job\_id AS "Job ID",l.city AS "City",SUM(e.salary) AS "Monthly Salary Cost"

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), -- Group by department and job title(l.city) -- Group by city)

ORDER BY "Department Name", "Job ID", "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 e.first\_name || ' ' || e.last\_name AS "Employee Name",e.department\_id AS "Department ID",NULL AS "Department Name",NULL AS "City"

FROM Employees e

UNION ALL

SELECT NULL AS "Employee Name",d.department\_id AS "Department ID",d.department\_name AS "Department Name",NULL AS "City"

FROM Departments d

UNION ALL

SELECT NULL AS "Employee Name",NULL AS "Department ID",NULL AS "Department Name",l.city AS "City"

FROM Locations l;

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 SUBSTR(e.first\_name, 1, 1) || ' ' || e.last\_name AS "Employee Name",e.salary AS "Salary",d.department\_name AS "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 d.department\_name, e.last\_name;