**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**

**SELECT table\_name**

**FROM user\_tables**

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

**Write the Query**

**• Problem: − Create a list that includes the first initial of every employee's first name, a space, and the last name of the employee**

**• Tables Used: − Employees**

**SELECT CONCAT(SUBSTR(first\_name, 1, 1), ' ', last\_name) AS employee\_name**

**FROM employees;**

**Write the Query**

**• 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'**

**• Tables Used: − Employees**

**SELECT first\_name || ' ' || last\_name AS full\_name,**

**email**

**FROM employees**

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

**Write the Query**

**• Problem: − Create a list of 'smallest' last name and the 'highest' last name from the employees table**

**• Tables Used: − Employees**

**SELECT MIN(last\_name) AS smallest\_last\_name,**

**MAX(last\_name) AS highest\_last\_name**

**FROM employees;**

**Write the Query**

**• 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**

**• Tables Used: − Employees**

**SELECT TO\_CHAR(salary, '$9999.99') AS formatted\_salary**

**FROM employees**

**WHERE salary BETWEEN 700 AND 3000;**

**Write the Query**

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

**• Tables Used: − Employees, Jobs**

**SELECT e.first\_name,**

**e.last\_name,**

**j.job\_title**

**FROM employees e**

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

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

**Write the Query**

**•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**

**•Tables Used: −Employees, Jobs**

**SELECT j.job\_title,**

**CONCAT(TO\_CHAR(j.min\_salary, '9999'), ' – ', TO\_CHAR(j.max\_salary, '9999')) AS salary\_range,**

**e.salary**

**FROM employees e**

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

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

**Write the Query**

**• 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**

**• Tables Used: − Employees, Departments**

**SELECT CONCAT(SUBSTR(e.first\_name, 1, 1), '.') AS first\_initial,**

**e.last\_name,**

**d.department\_name**

**FROM employees e**

**JOIN departments d ON e.department\_id = d.department\_id;**

**Write the Query**

**• Problem: − Change the previous listing to join only on the department\_idcolumn**

**• Tables Used: − Employees, Departments**

**SELECT CONCAT(SUBSTR(e.first\_name, 1, 1), '.') AS first\_initial,**

**e.last\_name,**

**d.department\_name**

**FROM employees e**

**JOIN departments d ON e.department\_id = d.department\_id;**

**Write the Query**

**• 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**

**• Tables Used: − Employees**

**SELECT last\_name,**

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

**FROM employees;**

**Write the Query**

**• 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**

**• QUERY: Write the Query Query Result: SELECT SUBSTR(first\_name,1 1)||' '|last\_name, "Employee Name", salary "Salary", DEC(commission\_pctNULL, 'No', 'Yes')'Commission' FROM employees**

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

**salary AS "Salary",**

**DECODE(commission\_pct, NULL, 'No', 'Yes') AS "Commission"**

**FROM employees;**

**Write the Query**

**• 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**

**• Tables Used: − Employees, Departments, Locations**

**SELECT d.department\_name,**

**e.last\_name,**

**l.city,**

**l.state\_province**

**FROM departments d**

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

**LEFT OUTER JOIN locations l ON d.location\_id = l.location\_id;**

**Write the Query**

**•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\_pctcolumn; if no commission, then display his manager\_id; if he has neither commission nor manager, then the number -1**

**•Tables Used: −Employees**

**SELECT first\_name,**

**last\_name,**

**COALESCE(commission\_pct, manager\_id, -1) AS "Commission/Manager/Default"**

**FROM employees;**

**Write the Query**

**• Problem: − Create a list of every employee's last name, salary, and job\_gradefor all employees working in departments with a department\_idgreater than 50**

**• Tables Used: − Employees, job\_grades**

**SELECT e.last\_name,**

**e.salary,**

**j.job\_grade**

**FROM employees e**

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

**WHERE e.department\_id > 50;**

**Write the Query**

**•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\_pctcolumn; if no commission, then display his manager\_id; if he has neither commission nor manager, then the number -1**

**•Tables Used: −Employees**

**SELECT first\_name,**

**last\_name,**

**COALESCE(commission\_pct, manager\_id, -1) AS "Commission/Manager/Default"**

**FROM employees;**

**Write the Query**

**• Problem: − Create a list of every employee's last name, salary, and job\_gradefor all employees working in departments with a department\_idgreater than 50**

**• Tables Used: − Employees, job\_grades**

**SELECT e.last\_name,**

**e.salary,**

**j.job\_grade**

**FROM employees e**

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

**WHERE e.department\_id > 50;**

**Write the Query**

**• Problem: − Produce a list of every employee's last name and department name − Include both employees without departments, and departments without employees**

**• Tables Used: − Employees, Departments**

**SELECT e.last\_name,**

**d.department\_name**

**FROM employees e**

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

**Write the Query**

**•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**

**•Tables Used: −Employees**

**SELECT LEVEL AS position,**

**e.last\_name AS employee\_last\_name,**

**m.last\_name AS manager\_last\_name**

**FROM employees e**

**LEFT JOIN employees m ON e.manager\_id = m.employee\_id**

**START WITH e.employee\_id = 100**

**CONNECT BY PRIOR e.employee\_id = e.manager\_id**

**ORDER BY LEVEL, e.last\_name;**

**Write the Query**

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

**• Tables Used: − Employees**

**SELECT**

**MIN(hire\_date) AS earliest\_hire\_date,**

**MAX(hire\_date) AS latest\_hire\_date,**

**COUNT(\*) AS number\_of\_employees**

**FROM employees;**

**Write the Query**

**• Problem: − Create a list of department names and the departmental costs (salaries added up) − Include only departments whose salary costs are between 1500 and 31000, and sort the listing by the cost**

**• Tables Used: − Employees, Departments**

**SELECT MIN(hire\_date) AS earliest\_hire\_date,**

**MAX(hire\_date) AS latest\_hire\_date,**

**COUNT(\*) AS number\_of\_employees**

**FROM employees;**

**Write the Query**

**• 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**

**• Tables Used: − Employees, Departments**

**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 1500 AND 31000**

**ORDER BY departmental\_cost;**

**Write the Query**

**• Problem: − Show the highest average salary for the departments in the employees table − Round the result to the nearest whole number**

**• Tables Used: − Employees**

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

**FROM (**

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

**FROM employees**

**GROUP BY department\_id**

**);**

**Write the Query**

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

**• Tables Used: − Employees, Departments**

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

**FROM employees e**

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

**GROUP BY d.department\_name;**

**Write the Query**

**• 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**

**• Tables Used: − Employees, Departments**

**SELECT**

**d.department\_name,**

**e.job\_id,**

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

**FROM employees e**

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

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

**UNION ALL**

**SELECT**

**'All Departments' AS department\_name,**

**e.job\_id,**

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

**FROM employees e**

**GROUP BY e.job\_id;**

**Write the Query**

**•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) •Tables Used: −Employees, Departments**

**SELECT**

**d.department\_name,**

**e.job\_id,**

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

**FROM employees e**

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

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

**Write the Query**

**• 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)**

**• Tables Used: − Employees, Departments**

**SELECT**

**d.department\_name,**

**e.job\_id,**

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

**CASE**

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

**WHEN GROUPING(e.job\_id) = 1 THEN 'Subtotal for All Job IDs'**

**ELSE 'Detailed Data'**

**END AS subtotal\_indicator**

**FROM employees e**

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

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

**Write the Query**

**• 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)**

**• Tables Used: − Employees, Departments, Locations**

**SELECT**

**d.department\_name,**

**e.job\_id,**

**l.city,**

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

**CASE**

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

**WHEN GROUPING(e.job\_id) = 1 THEN 'Subtotal for All Job Titles'**

**WHEN GROUPING(l.city) = 1 THEN 'Subtotal for All Cities'**

**ELSE 'Detailed Data'**

**END AS subtotal\_indicator**

**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)**

**);**

**Write the Query**

**•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)**

**•Tables Used: −Employees, Departments, Locations**

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

**SELECT**

**CONCAT(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**

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

**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**

**-- List of cities**

**SELECT**

**NULL AS employee\_name,**

**NULL AS department\_id,**

**NULL AS department\_name,**

**l.city AS city**

**FROM locations l;**

**Write the Query**

**• 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**

**• Tables Used: − Departments, Employees**

**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**

**);**