**Programming 6**

**Database Programming with SQL  
6-1: Cross Joins and Natural Joins  
Practice Activities**

Vocabulary  
Identify the vocabulary word for each definition below.

|  |  |
| --- | --- |
| **Cross Join** | Returns the Cartesian product from two tables. |
| **Natural Join** | Joins two tables based on the same column name. |

Try It / Solve It

1. Create a cross-join that displays the last name and department name from the employees and departments tables.

SELECT e.last\_name, d.department\_name

FROM employees e

CROSS JOIN departments d;

1. Create a query that uses a natural join to join the departments table and the locations table. Display the department id, department name, location id, and city.

SELECT department\_id, department\_name, location\_id, city

FROM departments

NATURAL JOIN locations;

1. Create a query that uses a natural join to join the departments table and the locations table. Restrict the output to only department IDs of 20 and 50. Display the department id, department name, location id, and city.

SELECT department\_id, department\_name, location\_id, city

FROM departments

NATURAL JOIN locations

WHERE department\_id IN (20, 50);

**Database Programming with SQL  
6-2: Join Clauses  
Practice Activities**

Vocabulary  
Identify the vocabulary word for each definition below.

|  |  |
| --- | --- |
| **USING clause** | Allows a natural join based on an arbitrary condition or two columns with different names. |
| **ON clause** | Performs an equijoin based on one specified column name |

Try It / Solve It

1. Join the Oracle database locations and departments table using the location\_id column. Limit the results to location 1400 only.

SELECT l.location\_id, l.city, d.department\_id, d.department\_name

FROM locations l

JOIN departments d ON l.location\_id = d.location\_id

WHERE l.location\_id = 1400;

1. Join DJs on Demand d\_play\_list\_items, d\_track\_listings, and d\_cds tables with the JOIN USING syntax. Include the song ID, CD number, title, and comments in the output.

SELECT song\_id, cd\_number, title, comments

FROM d\_play\_list\_items

JOIN d\_track\_listings USING (song\_id)

JOIN d\_cds USING (cd\_number);

1. Display the city, department name, location ID, and department ID for departments 10, 20, and 30 for the city of Seattle.

SELECT l.city, d.department\_name, d.location\_id, d.department\_id

FROM locations l

JOIN departments d ON l.location\_id = d.location\_id

WHERE l.city = 'Seattle' OR d.department\_id IN (10, 20, 30);

1. Display country name, region ID, and region name for Americas.

SELECT country\_name, regions.region\_id, region\_name

FROM countries

JOIN regions ON countries.region\_id = regions.region\_id

WHERE region\_name = 'Americas';

1. Write a statement joining the employees and jobs tables. Display the first and last names, hire date, job id, job title, and maximum salary. Limit the query to those employees who are in jobs that can earn more than $12,000.

SELECT e.first\_name, e.last\_name, e.hire\_date, e.job\_id, j.job\_title, j.max\_salary

FROM employees e

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

WHERE j.max\_salary > 12000;

1. Display job title, employee first name, last name, and email for all employees who are stock clerks.

SELECT j.job\_title, e.first\_name, e.last\_name, e.email

FROM employees e

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

WHERE j.job\_title = 'Stock Clerk';

1. Write a statement that displays the employee ID, first name, last name, manager ID, manager first  
   name, and manager last name for every employee in the employees table. Hint: this is a self-join.

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

FROM employees e

JOIN employees m ON e.manager\_id = m.employee\_id;

1. Use JOIN ON syntax to query and display the location ID, city, and department name for all Canadian locations.

SELECT l.location\_id, l.city, d.department\_name

FROM locations l

JOIN departments d ON l.location\_id = d.location\_id

WHERE l.country\_id = 'CA';

1. Query and display manager ID, department ID, department name, first name, and last name for all employees in departments 80, 90, 110, and 190.

SELECT e.manager\_id, d.department\_id, d.department\_name, e.first\_name, e.last\_name

FROM employees e

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

WHERE d.department\_id IN (80, 90, 110, 190);

1. Display employee ID, last name, department ID, department name, and hire date for those employees whose hire date was June 7, 1994.

SELECT e.employee\_id, e.last\_name, e.department\_id, d.department\_name, e.hire\_date

FROM employees e

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

WHERE e.hire\_date = TO\_DATE('07-JUN-1994', 'DD-MON-YYYY');

**Database Programming with SQL  
6-3: Inner versus Outer Joins  
Practice Activities**

Vocabulary  
Identify the vocabulary word for each definition below.

|  |  |
| --- | --- |
| **Full Outer Join** | Performs a join on two tables, retrieves all the rows in the Left table, even if there is no match in the Right table. It also retrieves all the rows in the Right table, even if there is no match in the Left table. |
| **Outer Join** | A join that returns the unmatched rows as well as matched rows |
| **Left Outer Join** | Performs a join on two tables, retrieves all the rows in the Left table even if there is no match in the Right table. |
| **Right Outer Join** | Performs a join on two tables, retrieves all the rows in the Right table even if there is no match in the Left table. |
| **Inner Join** | A join of two or more tables that returns only matched rows |

Try It / Solve It

1. Return the first name, last name, and department name for all employees including those employees not assigned to a department.

SELECT e.first\_name, e.last\_name, d.department\_name

FROM employees e

LEFT OUTER JOIN departments d

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

1. Return the first name, last name, and department name for all employees including those departments that do not have an employee assigned to them.

SELECT e.first\_name, e.last\_name, d.department\_name

FROM employees e

RIGHT OUTER JOIN departments d

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

1. Return the first name, last name, and department name for all employees including those departments that do not have an employee assigned to them and those employees not assigned to a department.

SELECT e.first\_name, e.last\_name, d.department\_name

FROM employees e

FULL OUTER JOIN departments d

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

1. Create a query of the DJs on Demand database to return the first name, last name, event date, and description of the event the client held. Include all the clients even if they have not had an event scheduled.

SELECT c.first\_name, c.last\_name, e.event\_date, e.description

FROM clients c

LEFT OUTER JOIN events e

ON c.client\_id = e.client\_id;

1. Using the Global Fast Foods database, show the shift description and shift assignment date even if there is no date assigned for each shift description.

SELECT s.shift\_description, a.assignment\_date

FROM shifts s

LEFT OUTER JOIN assignments a

ON s.shift\_id = a.shift\_id;

**Database Programming with SQL  
6-4: Self Joins and Hierarchical Queries  
Practice Activities**

Vocabulary  
Identify the vocabulary word for each definition below.

|  |  |
| --- | --- |
| **Self-Join** | Joins a table to itself |
| **Hierarchical Query** | Retrieves data based on a natural hierarchical relationship between rows in a table |
| **Level** | Determines the number of steps down from the beginning row that should be returned by a hierarchical query |
| **Start With** | Identifies the beginning row for hierarchical query |
| **Connect By** | Specifies the relationship between parent rows and child rows of a hierarchical query |

Try It / Solve It

1. Display the employee’s last name and employee number along with the manager’s last name and manager number. Label the columns: Employee, Emp#, Manager, and Mgr#, respectively.

SELECT e.last\_name AS Employee, e.employee\_id AS Emp#,

m.last\_name AS Manager, m.employee\_id AS Mgr#

FROM employees e

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

1. Modify question 1 to display all employees and their managers, even if the employee does not have a manager. Order the list alphabetically by the last name of the employee.

SELECT e.last\_name AS Employee, e.employee\_id AS Emp#,

NVL(m.last\_name, 'No Manager') AS Manager,

NVL(m.employee\_id, 'No Mgr#') AS Mgr#

FROM employees e

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

ORDER BY e.last\_name;

1. Display the names and hire dates for all employees who were hired before their managers, along with their managers’ names and hire dates. Label the columns Employee, Emp Hired, Manager and Mgr Hired, respectively.

SELECT e.last\_name AS Employee, e.hire\_date AS "Emp Hired",

m.last\_name AS Manager, m.hire\_date AS "Mgr Hired"

FROM employees e

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

WHERE e.hire\_date < m.hire\_date;

1. Write a report that shows the hierarchy for Lex De Haans department. Include last name, salary, and department id in the report.

SELECT LPAD(' ', LEVEL\*2, '-') || last\_name AS "Name", salary, department\_id

FROM employees

START WITH last\_name = 'De Haan'

CONNECT BY PRIOR employee\_id = manager\_id;

1. What is wrong in the following statement?  
   SELECT last\_name, department\_id, salary  
   FROM employees  
   START WITH last\_name = 'King'  
   CONNECT BY PRIOR manager\_id = employee\_id;

The CONNECT BY clause has the PRIOR keyword in the wrong place. It should reference the parent row, not the child row. The correct query should be:

SELECT last\_name, department\_id, salary

FROM employees

START WITH last\_name = 'King'

CONNECT BY PRIOR employee\_id = manager\_id;

1. Create a report that shows the organization chart for the entire employee table. Write the report so that each level will indent each employee 2 spaces. Since Oracle Application Express cannot display the spaces in front of the column, use - (minus) instead.

SELECT LPAD(' ', LEVEL\*2, '-') || last\_name AS "Name"

FROM employees

START WITH manager\_id IS NULL

CONNECT BY PRIOR employee\_id = manager\_id;

1. Re-write the report from 6 to exclude De Haan and all the people working for him.

SELECT LPAD(' ', LEVEL\*2, '-') || last\_name AS "Name"

FROM employees

START WITH manager\_id IS NULL

CONNECT BY PRIOR employee\_id = manager\_id

AND employee\_id != (SELECT employee\_id FROM employees WHERE last\_name = 'De Haan');