Database Programming with SQL  
6-1: Cross Joins and Natural Joins  
Vocabulary

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

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;**  
2. 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 d.department\_id, d.department\_name, l.location\_id, l.city**

**FROM departments d**

**NATURAL JOIN locations l;**  
3. 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 d.department\_id, d.department\_name, l.location\_id, l.city**

**FROM departments d**

**NATURAL JOIN locations l**

**WHERE d.department\_id IN (20, 50);**

6-2: Join Clauses  
Vocabulary

* **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

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

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

**FROM departments d**

**JOIN locations l**

**ON d.location\_id = l.location\_id**

**WHERE l.location\_id = 1400;**  
2. 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 dpli.song\_id, dtl.cd\_number, dtl.title, dtl.comments**

**FROM d\_play\_list\_items dpli**

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

**JOIN d\_cds dc USING (cd\_number);**  
3. 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 departments d**

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

**WHERE d.department\_id IN (10, 20, 30)**

**AND l.city = 'Seattle';**  
4. Display country name, region ID, and region name for Americas.

**SELECT c.country\_name, r.region\_id, r.region\_name**

**FROM countries c**

**JOIN regions r ON c.region\_id = r.region\_id**

**WHERE r.region\_name = 'Americas';**  
5. 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;**  
6. 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';**

The following questions use the JOIN...ON syntax:  
7. 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;**  
8. 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**

**JOIN countries c ON l.country\_id = c.country\_id**

**WHERE c.country\_name = 'Canada';**  
9. 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 e.department\_id IN (80, 90, 110, 190);**  
10. 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');**

6-3: Inner versus Outer Joins  
Vocabulary

* **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

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;**  
2. 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;**  
3. 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;**4. 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;**  
5. 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, sa.assignment\_date**

**FROM shifts s**

**LEFT OUTER JOIN shift\_assignments sa ON s.shift\_id = sa.shift\_id;**

6-4: Self Joins and Hierarchical Queries  
Vocabulary

* **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 a hierarchical query
* **CONNECT BY** - Specifies the relationship between parent rows and child rows of  
  a hierarchical query

For each problem, use the Oracle database.  
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**

**JOIN employees m ON e.manager\_id = m.employee\_id;**  
2. 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#, m.last\_name AS Manager, m.employee\_id AS Mgr#**

**FROM employees e**

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

**ORDER BY e.last\_name;**  
3. 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;**  
4. Write a report that shows the hierarchy for Lex De Haans department. Include last name, salary, and department id in the report.

**SELECT last\_name, salary, department\_id**

**FROM employees**

**START WITH last\_name = 'De Haan'**

**CONNECT BY PRIOR employee\_id = manager\_id;**  
5. 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 PRIOR clause should specify the parent-child relationship. The correct relationship is *CONNECT BY PRIOR employee\_id = manager\_id* (since the employee’s ID should match the manager’s ID from the previous level).**

6. 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('-', 2 \* (LEVEL - 1)) || last\_name AS "Employee", salary, department\_id**

**FROM employees**

**START WITH manager\_id IS NULL**

**CONNECT BY PRIOR employee\_id = manager\_id;**  
7. Re-write the report from 6 to exclude De Haan and all the people working for him

**SELECT LPAD('-', 2 \* (LEVEL - 1)) || last\_name AS "Employee", salary, department\_id**

**FROM employees**

**WHERE employee\_id NOT IN (**

**SELECT employee\_id**

**FROM employees**

**START WITH last\_name = 'De Haan'**

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

**)**

**START WITH manager\_id IS NULL**

**CONNECT BY PRIOR employee\_id = manager\_id;**