LAB 8: Displaying Data from Multiple Tables

Objectives

After completing this lesson, you should be able to do the following:

- ✓ Write SELECT statements to access data from more than one table using equality and nonequality joins
- ✓ View data that generally does not meet a join condition by using outer joins
- ✓ Join a table to itself by using a self join

Lesson Aim

This lesson covers obtaining data from more than one table.

Obtaining Data from Multiple Tables

Sometimes you need to use data from more than one table. In the example, the report displays data from two separate tables.

• Employee IDs exist in the EMPLOYEES table.

• Department IDs exist in both the EMPLOYEES and DEPARTMENTS tables.

• Location IDs exist in the DEPARTMENTS table.

To produce the report, you need to link the EMPLOYEES and DEPARTMENTS tables and access data from both of them.

Generating Cartesian Products

A Cartesian product is generated if a join condition is omitted. The example in the slide displays employee last name and department name from the `EMPLOYEES` and `DEPARTMENTS` tables. Because no `WHERE` clause has been specified, all rows (20 rows) from the `EMPLOYEES` table are joined with all rows (8 rows) in the `DEPARTMENTS` table, thereby generating 160 rows in the output.

```
SELECT last_name, department_name dept_name
FROM employees, departments;
```

Defining Joins

Using the SQL: 1999 syntax, you can obtain the same results as what was shown in the prior pages.
In the syntax:

`table1.column`	Denotes the table and column from which data is retrieved
`CROSS JOIN`	Returns a Cartesian product from the two tables
`NATURAL JOIN`	Joins two tables based on the same column name
`JOIN` *`table`* `USING` *`column_name`*	Performs an equijoin based on the column name
`JOIN` *`table ON table1.column_name = table2.column_name`*	Performs an equijoin based on the condition in the `ON` clause
`LEFT/RIGHT/FULL OUTER`	

Use a join to query data from more than one table.

```
SELECT table1.column, table2.column
FROM table1
[CROSS JOIN table2] |
[NATURAL JOIN table2] |
[JOIN table2 USING (column_name)] |
[JOIN table2
ON(table1.column_name = table2.column_name)] |
[LEFT|RIGHT|FULL OUTER JOIN table2
ON (table1.column_name = table2.column_name)];
```

Creating Cross Joins

```
SELECT last_name, department_name
FROM employees
CROSS JOIN departments;
```

Creating Natural Joins

It is possible to let the join be completed automatically based on columns in the two tables which have matching data types and names, using the keywords `NATURAL JOIN` keywords.
Note: The join can happen only on columns having the same names and data types in both the tables. If the columns have the same name, but different data types, then the `NATURAL JOIN` syntax causes an error.

```
SELECT department_id, department_name,
location_id, city
FROM departments
NATURAL JOIN locations;
```

```
SELECT department_id, department_name,
location_id, city
FROM departments
NATURAL JOIN locations
WHERE department_id IN (20, 50);
```

Creating Joins with the USING Clause

```
SELECT e.employee_id, e.last_name, d.location_id
FROM employees e JOIN departments d
USING (department_id);
```

The ON Condition

Use the `ON` clause to specify a join condition. Doing so lets you specify join conditions separate from any search or filter conditions in the `WHERE` clause.

Creating Joins with the ON Clause

The `ON` clause can also be used as follows to join columns that have different names:

```
SELECT e.last_name emp, m.last_name mgr
FROM employees e JOIN employees m
ON (e.manager_id = m.employee_id);
```

The preceding example is a self join of the `EMPLOYEE` table to itself, based on the `EMPLOYEE_ID` and `MANAGER_ID` columns.

Creating Three-Way Joins with the ON Clause

```
SELECT employee_id, city, department_name
FROM employees e
JOIN departments d
ON d.department_id = e.department_id
JOIN locations l
ON d.location_id = l.location_id;
```

• In SQL: 1999, the join of two tables returning only matched rows is an inner join.
• A join between two tables that returns the resultsof the inner join as well as unmatched rows left (or right) tables is a left (or right) outer join.
• A join between two tables that returns the results of an inner join as well as the results of a left and right join is a full outer join.

LEFT OUTER JOIN

Example of LEFT OUTER JOIN

This query retrieves all rows in the EMPLOYEES table, which is the left table even if there is no match in the DEPARTMENTS table.
This query was completed in earlier releases as follows

```
SELECT e.last_name, e.department_id, d.department_name
FROM employees e
LEFT OUTER JOIN departments d
ON (e.department_id = d.department_id);
```

RIGHT OUTER JOIN

Example of RIGHT OUTER JOIN

This query retrieves all rows in the DEPARTMENTS table, which is the right table even if there is no match in the EMPLOYEES table.
This query was completed in earlier releases as follows:

```
SELECT e.last_name, e.department_id, d.department_name
FROM employees e
RIGHT OUTER JOIN departments d
ON (e.department_id = d.department_id);
```

FULL OUTER JOIN

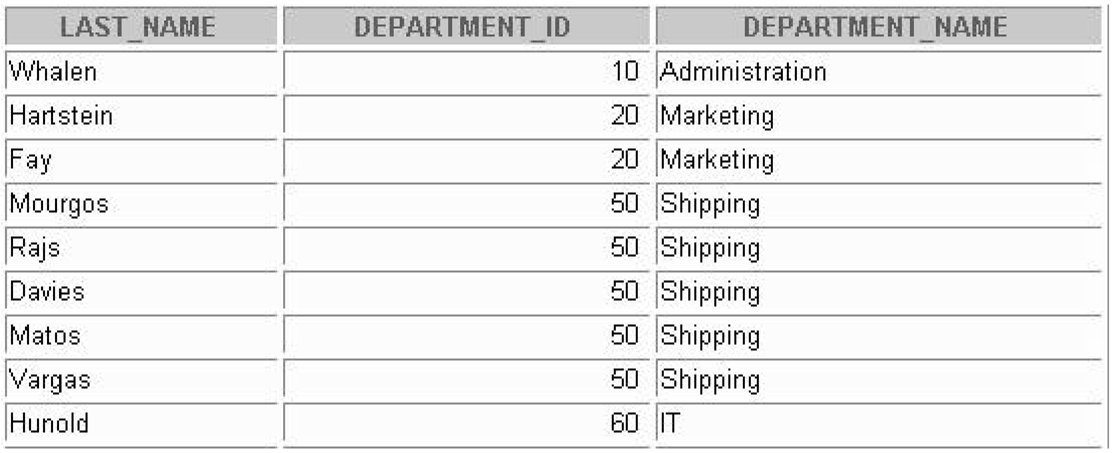
Example of FULL OUTER JOIN

This query retrieves all rows in the EMPLOYEES table, even if there is no match in the DEPARTMENTS table. It also retrieves all rows in the DEPARTMENTS table, even if there is no match in the EMPLOYEES table.

```
SELECT e.last_name, e.department_id, d.department_name
FROM employees e
FULL OUTER JOIN departments d
ON (e.department_id = d.department_id);
```

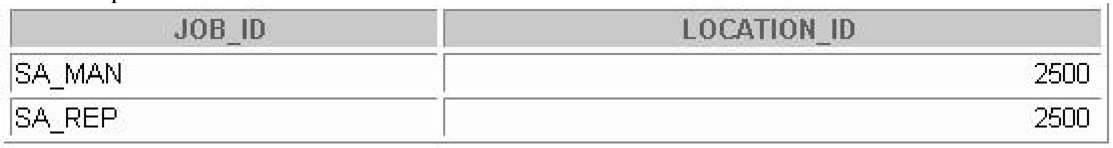
Practice

1. Write a query to display the last name, department number, and department name for all employees.



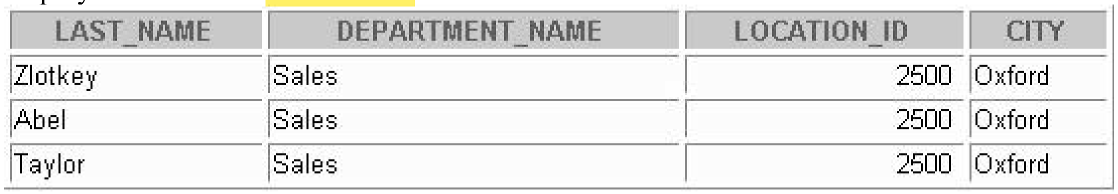
LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Hartstein	20	Marketing
Fay	20	Marketing
Mourgos	50	Shipping
Rajs	50	Shipping
Davies	50	Shipping
Matos	50	Shipping
Vargas	50	Shipping
Hunold	60	IT

2. Create a unique listing of all jobs that are in department 30. Include the location of department 90 in the output.



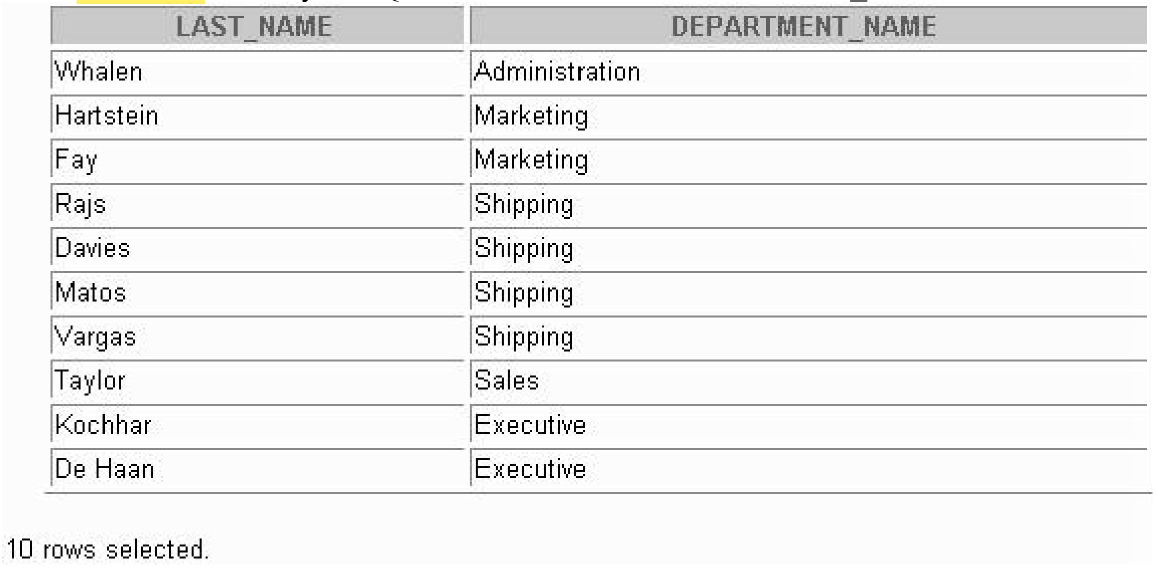
JOB_ID	LOCATION_ID
SA_MAN	2500
SA_REP	2500

3. Write a query to display the employee last name, department name, location ID, and city of all employees who earn a commission.



LAST_NAME	DEPARTMENT_NAME	LOCATION_ID	CITY
Zlotkey	Sales	2500	Oxford
Abel	Sales	2500	Oxford
Taylor	Sales	2500	Oxford

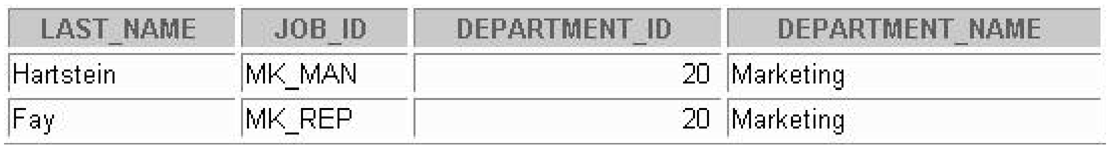
4. Display the employee last name and department name for all employees who have an *a* (lowercase) in their last names. Place your SQL statement in a text file named `lab4_4.sql`.



LAST_NAME	DEPARTMENT_NAME
Whalen	Administration
Hartstein	Marketing
Fay	Marketing
Rajs	Shipping
Davies	Shipping
Matos	Shipping
Vargas	Shipping
Taylor	Sales
Kochhar	Executive
De Haan	Executive

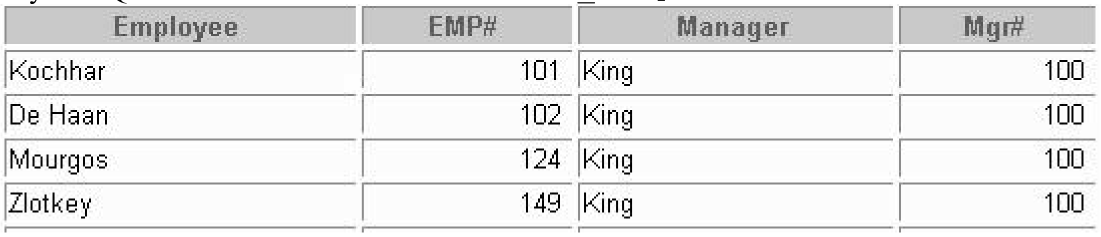
```
10 rows selected.
```

5. Write a query to display the last name, job, department number, and department name for all employees who work in Toronto.



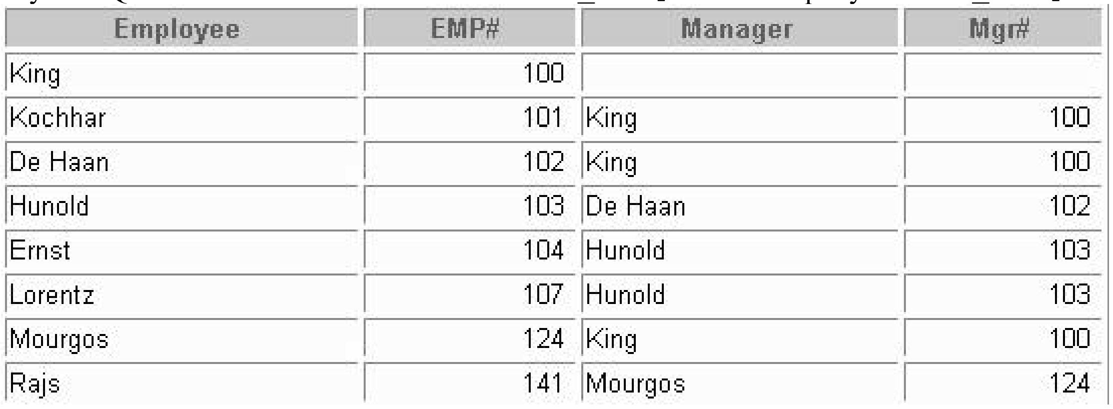
LAST_NAME	JOB_ID	DEPARTMENT_ID	DEPARTMENT_NAME
Hartstein	MK_MAN	20	Marketing
Fay	MK_REP	20	Marketing

6. Display the employee last name and employee number along with their manager's last name and manager number. Label the columns `Employee`, `Emp#`, `Manager`, and `Mgr#`, respectively.
Place your SQL statement in a text file named `lab4_6.sql`.



Employee	EMP#	Manager	Mgr#
Kochhar	101	King	100
De Haan	102	King	100
Mourgos	124	King	100
Zlotkey	149	King	100

7. Modify `lab4_6.sql` to display all employees including King, who has no manager. Order the results by the employee number.
Place your SQL statement in a text file named `lab4_7.sql`. Run the query in `lab4_7.sql`.



Employee	EMP#	Manager	Mgr#
King	100		
Kochhar	101	King	100
De Haan	102	King	100
Hunold	103	De Haan	102
Ernst	104	Hunold	103
Lorentz	107	Hunold	103
Mourgos	124	King	100
Rajs	141	Mourgos	124