Lab 02: Querying Database Tables

Objective(s):

To learn the WHERE and ORDER BY clauses in SELECT statement.

1. WHERE Clause

The next thing we want to do is to start limiting, or filtering, the data we fetch from the database. By adding a WHERE clause to the SELECT statement, we add one (or more) conditions that must be met by the selected data. This will limit the number of rows that answer the query and are fetched. In many cases, this is where most of the "action" of a query takes place.

We can continue with our previous query, and limit it to only those employees living in London:

```
SELECT EmployeeID, FirstName, LastName, HireDate, City FROM Employees

WHERE City = 'London'
```

Resulting in

EmployeeID	FirstName	LastName	HireDate	City
5	Steven	Buchanan	17/10/1993 12:00:00 AM	London
6	Michael	Suyama	17/10/1993 12:00:00 AM	London
7	Robert	King	2/1/1994 12:00:00 AM	London
9	Anne	Dodsworth	15/11/1994 12:00:00 AM	London

If you wanted to get the opposite, the employees who do not live in London, you would write

```
SELECT EmployeeID, FirstName, LastName, HireDate, City FROM Employees

WHERE City <> 'London'
```

It is not necessary to test for equality; you can also use the standard equality/inequality operators that you would expect. For example, to get a list of employees who were hired on or after a given date, you would write

```
SELECT EmployeeID, FirstName, LastName, HireDate, City FROM Employees

WHERE HireDate >= '1-july-1993'
```

and get the resulting rows

EmployeeID	FirstName	LastName	HireDate	City
5	Steven	Buchanan	17/10/1993 12:00:00 AM	London
6	Michael	Suyama	17/10/1993 12:00:00 AM	London
7	Robert	King	2/1/1994 12:00:00 AM	London
8	Laura	Callahan	5/3/1994 12:00:00 AM	Seattle
9	Anne	Dodsworth	15/11/1994 12:00:00 AM	London

Of course, we can write more complex conditions. The obvious way to do this is by having multiple conditions in the WHERE clause. If we want to know which employees were hired between two given dates, we could write

```
SELECT EmployeeID, FirstName, LastName, HireDate, City

FROM Employees

WHERE (HireDate >= '1-june-1992') AND (HireDate <= '15-december-1993')
```

resulting in

EmployeeID	FirstName	LastName	HireDate	City
2	Andrew	Fuller	14/8/1992 12:00:00 AM	Tacoma
4	Margaret	Peacock	3/5/1993 12:00:00 AM	Redmond
5	Steven	Buchanan	17/10/1993 12:00:00 AM	London
6	Michael	Suyama	17/10/1993 12:00:00 AM	London

Note that SQL also has a special BETWEEN operator that checks to see if a value is between two values (including equality on both ends). This allows us to rewrite the previous query as

```
SELECT EmployeeID, FirstName, LastName, HireDate, City

FROM Employees

WHERE HireDate BETWEEN '1-june-1992' AND '15-december-1993'
```

We could also use the NOT operator, to fetch those rows that are not between the specified dates:

```
SELECT EmployeeID, FirstName, LastName, HireDate, City

FROM Employees

WHERE HireDate NOT BETWEEN '1-june-1992' AND '15-december-1993'
```

If we want to check if a column value is equal to more than one value? If it is only two values, then it is easy enough to test for each of those values, combining them with the OR operator and writing something like

```
SELECT EmployeeID, FirstName, LastName, HireDate, City FROM Employees

WHERE City = 'London' OR City = 'Seattle'
```

However, if there are three, four, or more values that we want to compare against, the above approach quickly becomes messy. In such cases, we can use the IN operator to test against a set of values. If we wanted to see if the City was either Seattle, Tacoma, or Redmond, we would write

```
SELECT EmployeeID, FirstName, LastName, HireDate, City FROM Employees

WHERE City IN ('Seattle', 'Tacoma', 'Redmond')
```

producing the results shown below.

EmployeeID	FirstName	LastName	HireDate	City
1	Nancy	Davolio	1/5/1992 12:00:00 AM	Seattle
2	Andrew	Fuller	14/8/1992 12:00:00 AM	Tacoma
4	Margaret	Peacock	3/5/1993 12:00:00 AM	Redmond
8	Laura	Callahan	5/3/1994 12:00:00 AM	Seattle

As with the BETWEEN operator, here too we can reverse the results obtained and query for those rows where City is not in the specified list:

```
SELECT EmployeeID, FirstName, LastName, HireDate, City FROM Employees

WHERE City NOT IN ('Seattle', 'Tacoma', 'Redmond')
```

Finally, the LIKE operator allows us to perform basic pattern-matching using wildcard characters. For Microsoft SQL Server, the wildcard characters are defined as follows:

Wildcard	Description
_ (underscore)	matches any single character
%	matches a string of zero or more characters
[ ]	matches any single character within the specified range (e.g. [a-f]) or set (e.g. [abcdef]).
[^]	matches any single character not within the specified range (e.g. [^a-f]) or set (e.g. [^abcdef]).

A few examples should help clarify these rules.

- **WHERE** FirstName **LIKE** '_im' finds all three-letter first names that end with 'im' (e.g. Jim, Tim).
- **WHERE** LastName **LIKE** '%stein' finds all employees whose last name ends with 'stein'
- **WHERE** LastName **LIKE** '%stein%' finds all employees whose last name includes 'stein' anywhere in the name.
- **WHERE** FirstName **LIKE** '[JT]im' finds three-letter first names that end with 'im' and begin with either 'J' or 'T' (that is, only Jim and Tim)
- **WHERE** LastName **LIKE** 'm[^c]%' finds all last names beginning with 'm' where the following (second) letter is not 'c'.

Here too, we can opt to use the NOT operator: to find all of the employees whose first name does not start with 'M' or 'A', we would write

```
SELECT EmployeeID, FirstName, LastName, HireDate, City FROM Employees
WHERE (FirstName NOT LIKE 'M%') AND (FirstName NOT LIKE 'A%')
```

resulting in

EmployeeID	FirstName	LastName	HireDate	City
1	Nancy	Davolio	1/5/1992 12:00:00 AM	Seattle
3	Janet	Leverling	1/4/1992 12:00:00 AM	Kirkland
5	Steven	Buchanan	17/10/1993 12:00:00 AM	London
7	Robert	King	2/1/1994 12:00:00 AM	London
8	Laura	Callahan	5/3/1994 12:00:00 AM	Seattle

1. Order By Clause

Until now, we have been discussing filtering the data: that is, defining the conditions that determine which rows will be included in the final set of rows to be fetched and returned from the database. Once we have determined which columns and rows will be included in the results of our SELECT query, we may want to control the order in which the rows appear—sorting the data.

To sort the data rows, we include the ORDER BY clause. The ORDER BY clause includes one or more column names that specify the sort order. If we return to one of our first SELECT statements, we can sort its results by City with the following statement:

```
SELECT EmployeeID, FirstName, LastName, HireDate, City FROM Employees

ORDER BY City
```

By default, the sort order for a column is ascending (from lowest value to highest value), as shown below for the previous query:

EmployeeID	FirstName	LastName	HireDate	City
3	Janet	Leverling	1/4/1992 12:00:00 AM	Kirkland
5	Steven	Buchanan	17/10/1993 12:00:00 AM	London
6	Michael	Suyama	17/10/1993 12:00:00 AM	London
7	Robert	King	2/1/1994 12:00:00 AM	London
9	Anne	Dodsworth	15/11/1994 12:00:00 AM	London
4	Margaret	Peacock	3/5/1993 12:00:00 AM	Redmond
1	Nancy	Davolio	1/5/1992 12:00:00 AM	Seattle
8	Laura	Callahan	5/3/1994 12:00:00 AM	Seattle
2	Andrew	Fuller	14/8/1992 12:00:00 AM	Tacoma

If we want the sort order for a column to be descending, we can include the DESC keyword after the column name.

The ORDER BY clause is not limited to a single column. You can include a comma-delimited list of columns to sort by—the rows will all be sorted by the first column specified and then by the next column specified. If we add the Country field to the SELECT clause and want to sort by Country and City, we would write:

SELECT EmployeeID, FirstName, LastName, HireDate, Country, City FROM Employees

ORDER BY Country, City DESC

Note that to make it interesting, we have specified the sort order for the City column to be descending (from highest to lowest value). The sort order for the Country column is still ascending. We could be more explicit about this by writing

SELECT EmployeeID, FirstName, LastName, HireDate, Country, City FROM Employees

ORDER BY Country ASC, City DESC

but this is not necessary and is rarely done. The results returned by this query are

EmployeeID	FirstName	LastName	HireDate	Country	City
5	Steven	Buchanan	17/10/1993 12:00:00 AM	UK	London
6	Michael	Suyama	17/10/1993 12:00:00 AM	UK	London
7	Robert	King	2/1/1994 12:00:00 AM	UK	London
9	Anne	Dodsworth	15/11/1994 12:00:00 AM	UK	London
2	Andrew	Fuller	14/8/1992 12:00:00 AM	USA	Tacoma
1	Nancy	Davolio	1/5/1992 12:00:00 AM	USA	Seattle
8	Laura	Callahan	5/3/1994 12:00:00 AM	USA	Seattle
4	Margaret	Peacock	3/5/1993 12:00:00 AM	USA	Redmond
3	Janet	Leverling	1/4/1992 12:00:00 AM	USA	Kirkland

It is important to note that a column does not need to be included in the list of selected (returned) columns in order to be used in the ORDER BY clause. If we don't need to see/use the Country values, but are only interested in them as the primary sorting field we could write the query as

SELECT EmployeeID, FirstName, LastName, HireDate, City FROM Employees

ORDER BY Country ASC, City DESC

with the results being sorted in the same order as before:

EmployeeID	FirstName	LastName	HireDate	City
5	Steven	Buchanan	17/10/1993 12:00:00 AM	London
6	Michael	Suyama	17/10/1993 12:00:00 AM	London
7	Robert	King	2/1/1994 12:00:00 AM	London
9	Anne	Dodsworth	15/11/1994 12:00:00 AM	London
2	Andrew	Fuller	14/8/1992 12:00:00 AM	Tacoma
1	Nancy	Davolio	1/5/1992 12:00:00 AM	Seattle
8	Laura	Callahan	5/3/1994 12:00:00 AM	Seattle
4	Margaret	Peacock	3/5/1993 12:00:00 AM	Redmond
3	Janet	Leverling	1/4/1992 12:00:00 AM	Kirkland

Exercise

Now by using the EMP table, formulate the following queries in SQL

1. Display empno,sal of employees whose sal is less than 2000.

2. List ename of all employees whose name starts with letter 'A'.

3. Display ename,job,deptno of employees who are either SALESMAN or belongs to deptno = 10.

4. Display ename,sal of employees whose salaries lies in the range of 1500 to 3000.

5. Display ename,comm of employees whose commission is empty.

6. Display names of all employees that end with alphabet 'N'.

7. Show ename,sal,job of those employees who are not 'SALESMAN'.

8. Show empno,sal of those employees who do not have salaries of 1600, 1250, 3000.

9. Show names of those employees whose name starts with A and ends with N.

10. Display the list of employee names that have letters 'LA' in their names.

11. Display the empno,sal of employees. In that, the highest paid employee should display first and lowest paid should display last.