Bangladesh University of Business and Technology
Department of Computer Science and Engineering

CSE 232: Database Systems Lab

Lab 03 Tasks - WHERE, UPDATE, DELETE

The WHERE Clause

The where clause allows us to select only those rows in the result relation of the FROM clause that satisfy a specified condition.
The basic syntax involving the WHERE clause is as follows:

```
SELECT column1, column2
FROM table_name
WHERE condition;
```

SQL allows us to specify multiple conditions as well by using logical operators. For example, the query "Find the names of all instructors in the Computer Science department who have salary greater than $70,000." can be written as:

```
SELECT name
FROM instructor
WHERE dept_name = 'Comp. Sci.' AND salary > 70000;
```

A list of operators supported in SQL can be found here.

AND, OR, and NOT

The WHERE clause can be combined with AND, OR, and NOT operators.
The AND and OR operators are used to filter records based on more than one condition:

- The AND operator displays a record if all the conditions separated by AND are TRUE.
- The OR operator displays a record if any of the conditions separated by OR is TRUE.

The NOT operator displays a record if the condition(s) is NOT TRUE.

```
--AND--
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;

--OR--
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;

--NOT--
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

The IN Operator

The IN operator allows you to specify multiple values in a WHERE clause.
The IN operator is a shorthand for multiple OR conditions.

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);

--Example--
SELECT * FROM Customers
WHERE Country IN ('Germany', 'France', 'UK');
```

It is also possible to imply conditions that rely on other relations by using sub-queries inside the WHERE clause.
For example, the following SQL statement selects all customers that are from the same countries as the suppliers:

```
SELECT * FROM Customers
WHERE Country IN (SELECT Country FROM Suppliers);
```

The BETWEEN Operator

The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.
The BETWEEN operator is inclusive: begin and end values are included.

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;

--Example--
SELECT * FROM Products
WHERE Price NOT BETWEEN 10 AND 20;

SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20
AND NOT CategoryID IN (1,2,3);

SELECT * FROM Products
WHERE ProductName NOT BETWEEN 'Garden Ramsay' AND 'Nike di Giovanni';
```

The ORDER BY Clause

The ORDER BY keyword is used to sort the result-set in ascending or descending order.
The ORDER BY keyword sorts the records in ascending order by default. Otherwise, to specify ascending or descending order of sorting, we use the keywords 'ASC' and 'DESC' respectively.

```
--Ascending--
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC;
--Descending--
SELECT column1, column2, ...
FROM table_name
```

```
ORDER BY column1, column2, ... DESC;
--Example--
SELECT * FROM Customers
ORDER BY Country, CustomerName;

SELECT * FROM Customers
ORDER BY Country ASC, CustomerName DESC;
```

The UPDATE Statement

The UPDATE statement is used to modify the existing records in a table.

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;

--Example--
UPDATE Customers
SET ContactName = 'Albert Samuels', City= 'Venice'
WHERE CustomerID = 1;
```

Warning: Be careful when updating records. If you omit the WHERE clause, ALL records will be updated!

The DELETE Statement

The DELETE statement is used to delete existing records in a table.

```
DELETE FROM table_name WHERE condition;

--Example--

DELETE FROM Customers WHERE CustomerName='Abegail Lamborgini';
```

Warning: Similar to the UPDATE statement, not specifying a WHERE clause will result in deleting all the rows.

The SELECT DISTINCT Statement

The SELECT DISTINCT statement is used to return only distinct (different) values.
Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

```
SELECT DISTINCT column1, column2, ...
FROM table_name;

--Example--

SELECT DISTINCT Country FROM Customers;
```

Aliases

SQL aliases are used to give a table, or a column in a table, a temporary name.
Aliases are often used to make column names more readable.
An alias only exists for the duration of the query.

```
SELECT column_name AS alias_name
FROM table_name;

--Example--

SELECT column_name(s)
FROM table_name AS alias_name;
```

Tasks

Run the codes in Citizen.txt

1. Show all the values of the citizen table.
2. Show only the c_name, age and occupation from the table.
3. Show the names of the citizens who are living in Dhaka.
4. Show the names and hometowns of those whose age is less than 45.
5. Make a list of people (all attributes) whose income is within 50,000/- and 90,000/-.
6. Make a list (all attributes) of the female citizens.
7. Make a list (all attributes) of engineers, doctors and retired citizens.
8. Show the ID, name and salary of musicians and businessmen.
9. Show the list of citizens whose occupations might be either doctor or engineer.
10. Make an ordered list of engineers according to their salary.
11. Make a descending-ordered list based on age and only show the names and age. The names should be displayed in ascending order if the ages are the same.
12. Make a list of all the distinct c_home values in the tables. Rename the output column as 'Unique_District'
13. Create a new table –
    Updated_Citizen(C_ID, Name, C_Home, Age, Occupation, Gender, New_Salary).
    Populate the table with the same entries as the 'Citizen' table. Now, your task is to change the value of C_Home from 'Ctg' to 'Chittagong' wherever applicable.
14. Delete the top 10 rows from the 'Updated_Citizen' table.
15. Increase the salary by 30% in the 'Updated_Citizen' table.