GIFT School of Engineering and Applied Sciences

Fall 2022

CS-244: Database Systems-Lab

Lab-06 Manual

Data Selection and Filtering using SELECT Statement

Introduction to Lab

This lab will explain the usage of the SELECT statement for data selection and data filtering. Is also introduces the usage of common arithmetic function using the SELECT statement. Finally, students are introduced to the usage of WHERE clause for the selection of data using the SELECT statement.

The main topics of this lab include:

1. Introduction to Structured Query Language (SQL)
2. Capabilities of SQL SELECT Statements
3. Basic SELECT Statement
4. Selecting ALL Columns
5. Selecting Specific Columns
6. Writing SELECT Statements
7. Column Heading Defaults
8. Arithmetic Expressions
9. Using Arithmetic Operators
10. Operator Precedence & Parenthesis
11. Column Aliases
12. Null Values
13. Concat()
14. Literal Character Strings
15. Eliminating Duplicate Rows
16. Limiting Rows Using Selection: the WHERE Clause
17. Character Strings and Dates
18. Comparison Conditions (=, >, <, >=, <=, <>, BETWEEN, IN, LIKE, IS NULL)
19. String Wildcards (%, _)
20. Using the NULL Conditions
21. Logical Conditions (AND, OR, NOT)
22. ORDER BY Clause
23. Practice SQL Statements
24. SOLUTIONS: Practice SQL Statements

Objectives of this Lab

At the end of this lab, students should be able to:

a) Understand the usage of SELECT statement for displaying selected rows

b) Understand the usage of various conditional and logical operators in the
    i. SELECT statement.
    ii. Understand the usage of simple string matching using LIKE and string wildcards
c) Understand how to use the SQL SELECT statement for creating queries having multiple selection criteria
d) Understand how to display data in sorted order using the SELECT statement

1. Introduction to Structured Query Language (SQL)

i. SQL stands for Structured Query Language, which is a standardized language for interacting with RDBMS (Relational Database Management System). Some of the popular relational database example are: MySQL, Oracle, mariaDB, postgreSQL etc.

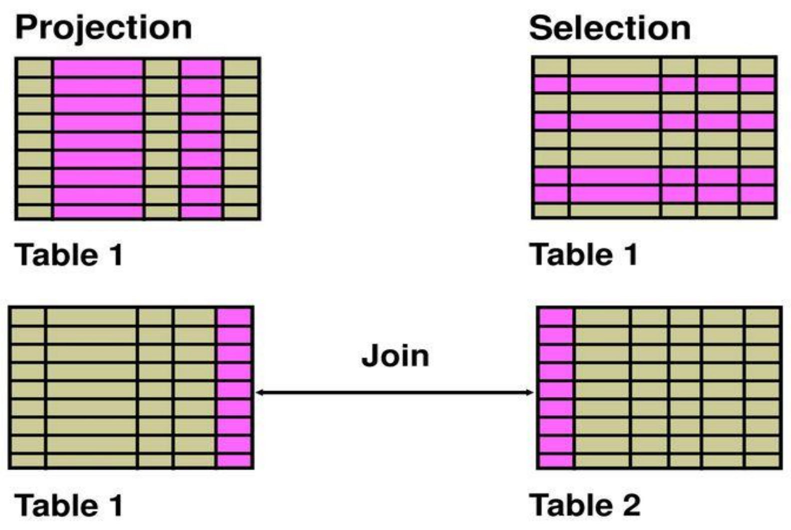
ii. SQL is used to perform C.R.U.D (Create, Retrieve, Update & Delete) operations on relational databases.

iii. SQL can also perform administrative tasks on database such as database security, backup, user management etc.

iv. We can create databases and tables inside database using SQL.

2. Capabilities of SQL SELECT Statements

The SELECT statement is the most frequently used statement in SQL; it is used to retrieve data from one or more tables. The names of the tables used in the statement must be listed in the statement.



Using a SELECT statement, you can do the following:

- **Projection:** Choose the columns in a table that are returned by a query. Choose as few or as many of the columns as needed

- **Selection:** Choose the rows in a table that are returned by a query. Various criteria can be used to restrict the rows that are retrieved.

- **Joining:** Bring together data that is stored in different tables by specifying the link between them. SQL joins are covered in more detail in a later lab.

3. Basic SELECT Statement

SELECT *|{[DISTINCT] ***column*** **|*expression***
[*alias*],...} FROM ***table;***

In its simplest form, a **SELECT** statement must include the following:

- A **SELECT** clause, which specifies the columns to be displayed
- A **FROM** clause, which identifies the table containing the columns that are listed in the **SELECT** clause

In the syntax:

SELECT	**is a list of one or more columns**
*****	selects all columns

DISTINCT	suppresses duplicates
Column|expression	selects the named column or the expression
alias	gives selected columns different headings
FROM *table*	Specifies the table containing the columns

Note: Throughout these labs, the words *keyword*, *clause*, and *statement* are used as follows:

- A *keyword* refers to an individual SQL element.
  For example, **SELECT** and **FROM** are keywords.
- A *clause* is a part of a SQL statement.
  For example, **SELECT employee_id, last_name, ...** is a clause.
- A *statement* is a combination of two or more clauses.
  For example, **SELECT * FROM employees** are a SQL statement.

4. Selecting ALL Columns

SELECT * FROM dept;

You can display all columns of data in a table by following the SELECT keyword with an asterisk (*) You can also display all columns in the table by listing all the columns after the SELECT keyword.

5. Selecting Specific Columns

You can use the SELECT statement to display specific columns of the table by specifying the column names, separated by commas. This example below displays all the employee numbers, employee names, and salaries from the EMP table.

SELECT empno, ename, sal From emp;

Note: In the SELECT clause, specify the columns that you want, in the order in which you want them to appear in the output.

6. Writing SELECT Statements

Using the following simple rules and guidelines, you can construct valid statements that are both easy to read and easy to edit:

- SQL statements are not case-sensitive (unless indicated).
- SQL statements can be entered on one or many lines.
- Keywords cannot be split across lines or abbreviated.

- Clauses are usually placed on separate lines for readability and ease of editing.
- Indents should be used to make code more readable.
- Keywords typically are entered in uppercase; all other words, such as table names and columns, are entered in lowercase.

7. Column Heading Defaults

The column names are used for the default headings. For example,

```
SELECT empno, ename, job FROM emp;
```

8. Arithmetic Expressions

You may need to modify the way in which data is displayed, or you may want to perform calculations or look at what-if scenarios. These are all possible using arithmetic expressions. An arithmetic expression can contain column names, constant numeric values, and the arithmetic operators.

Arithmetic Operators

The following arithmetic operators are available in SQL. You can use arithmetic operators in any clause of a SQL statement (except the **FROM** clause).

Operator	**Description**
+	Add
-	Subtract
*	Multiply
/	Divide

Note: With DATE and TIMESTAMP data types, you can use the addition and subtraction operators only.

9. Using Arithmetic Operators

You can create expressions using date and numeric data using mathematical operators (**+**, **-**, *****, /).

```
SELECT ename, sal, sal + 300 FROM emp;
```

This example uses the addition operator to calculate a salary increase of $300 for all employees.This example also displays a SAL+300 column in the output.Note that the resultant calculated column SAL+300 is not a new column in the EMP table; it is for display only. By default, the name of a new column comes from the calculation that generated it—in this case, sal+300.

10. Operator Precedence & Parenthesis

In mathematical expressions, multiplication and division takes priority over addition and subtraction. If operators within an expression are of same priority, then evaluation is done from left to right. You can use parenthesis to force the expression within parenthesis to be evaluated first.

```
SELECT ename, sal, 12*sal+100 FROM emp;
```

You can override the rules of precedence by using parentheses to specify the desired order in which operators are to be executed.

```
SELECT ename, sal, 12*(sal+100) FROM emp;
```

Rules of Precedence:

- Multiplication and division occur before addition and subtraction.
- Operators of the same priority are evaluated from left to right.
- Parentheses are used to override the default precedence or to clarify the statement

11. Column Alias

You can override a column heading display with a column alias. It renames a column heading into something that is specified by the user. It is useful for reporting and calculations. The alias immediately follows the column name and is enclosed in double quotation marks. There could also be an optional **AS** keyword between the column name and the alias name. For example,

```
1. SELECT ename AS "Employee Name"
   FROM emp;
2. SELECT ename "Employee Name", mgr "Manager"
   FROM emp;
3. SELECT ename "Employee Name", sal*12 "Annual Salary"
   FROM emp;
```

12. Null Values

A null is a value that is unavailable, unassigned, unknown, or inapplicable. If a column is missing a value, then that column is said to be null, or contains null. A null is not the same as a zero or a space. A zero is a number, and a space is a character.

SELECT ename, sal, comm FROM emp;

Applying a mathematical operator on a null value does not change its value.

SELECT ename, sal, comm + 100 AS "New Commission" FROM emp;

Note: Columns of any data type can contain nulls. However, some constraints (NOT NULL and PRIMARY KEY) prevent nulls from being used in the column.

In the COMM column in the EMP table, notice that only a sales manager or sales representative can earn a commission. Other employees are not entitled to earn commissions. A null represents that fact.

13. Concatenation

You can link columns to other columns, arithmetic expressions, or constant values to create a character expression by using the concatenation operator (||). Columns on either side of the operator are combined to make a single output column.

1. **SELECT CONCAT (ename,job) FROM emp;**

Null Values in concatenation

If you concatenate a null value with a character string, the result is a character string.

ENAME, NULL results in Null.

14. Literal Character Strings

A literal value is a character, a number, or a date that is included in the SELECT list and that is not a column name or a column alias. It is printed for each row returned. Date and character literals must be enclosed in single quotation marks (' '). Number literals need not.

1. **SELECT CONCAT("Employee " , ename , ' works as a ' , job) "Employee Details" FROM emp;**
2. **SELECT CONCAT('Employee ' ,ename , ' was hired on ' , hiredate ,' is working as , job , ' and is getting ' , sal) AS**

15. Eliminating Duplicate Rows

The default display of queries is all rows, including duplicate rows.

```
SELECT deptno FROM emp;
```

To eliminate duplicate rows in the result, include the DISTINCT keyword in the SELECT clause immediately after the SELECT keyword.

```
SELECT DISTINCT deptno FROM emp;
```

You can specify multiple columns after the DISTINCT qualifier. The DISTINCT qualifier affects all the selected columns, and the result is every distinct combination of the columns.

```
SELECT DISTINCT deptno, job FROM emp;
```

16. Limiting Rows Using Selection: the WHERE Clause

You can restrict the rows returned from the query by using the WHERE clause. A WHERE clause contains a condition that must be met, and it directly follows the FROM clause. If the condition is true, the row meeting the condition is returned.

```
SELECT <criteria> FROM <table> [WHERE condition(s)];
```

The WHERE clause can compare values in columns, literal values, arithmetic expressions, or functions.

```
SELECT empno, ename, job, deptno FROM emp WHERE deptno = 30;
```

17. Character Strings and Dates

Character strings and dates in the WHERE clause must be enclosed in single quotation marks (“). Number constants should not be enclosed in single quotation marks. All character searches are case sensitive.

```
SELECT empno, job, deptno FROM emp WHERE ename = ‘WARD’;
SELECT * FROM emp WHERE hiredate = ’1980-12-17’;
SELECT * FROM emP WHERE ename = ‘king’;
SELECT empno, ename, job, deptno FROM emp WHERE deptno = 30;
```

18. Conditions

Comparison conditions are used in conditions that compare one expression to another value or expression.

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than

`<=`	Less than or equal to
`<>, !=, ^=`	Not equal to
`BETWEEN … AND …`	Between two values inclusive
`IN (set)`	Match any of a list a values
`LIKE`	Match a character pattern
`IS NULL`	Is a null value

```
1. SELECT empno, ename, job, sal FROM emp WHERE sal <= 3000;
2. SELECT * FROM emp WHERE hiredate > '1980-12-17';
3. SELECT ename, sal  FROM emp WHERE sal BETWEEN 2500 AND 3500;
4. SELECT empno, ename, sal ,deptno FROM emp WHERE mgr IN (7698, 7566
   , 7782);
5. SELECT empno, ename FROM emp WHERE ename between 'KING' and
   'SMITH';
6. SELECT empno, ename  FROM emp WHERE ename IN ('KING', 'SMITH');
```

Note:

- An alias cannot be used in the WHERE clause.
- Values that are specified with the BETWEEN condition are inclusive. You must specify the lower limit first.
- You can also use the BETWEEN condition on character values.
- The IN condition can be used with any data type.

19. String Wildcards (%, _)

You may select rows that match a character pattern by using the LIKE condition. The character pattern-matching operation is referred to as wildcard search. Two symbols can be used to construct the search string:

Symbol	Description
%	Represents any sequence of zero or more characters
_	Represents any single character

```
1. SELECT empno, ename, job, sal FROM emp WHERE ename LIKE 'S%';
2. SELECT empno, ename, job, sal FROM emp WHERE ename LIKE '_I%';
3. SELCT empno, ename, job, sal FROM emp WHERE ename LIKE '%N';
```

The LIKE condition can be used as a shortcut for some BETWEEN comparisons. The following example displays the last names and hire dates of all employees who joined between January 1995 and December 1995:

```
SELECT ename, hiredate FROM emp WHERE hiredate LIKE '%95';
```

20. Using the NULL Conditions

The NULL conditions include the **IS NULL** condition and the **IS NOT NULL** condition. The **IS NULL** condition tests for nulls. We cannot use a **=** to test nulls.

```
1. SELECT empno, ename, job, sal FROM emp WHERE mgr IS NULL;
2. SELECT empno, ename, job, sal FROM emp WHERE comm IS NOT NULL;
3. SELECT empno, ename, sal, deptno FROM emp WHERE comm IS NULL;
4. SELECT empno, ename, sal, deptno FROM emp WHERE comm IS NOT NULL;
```

21. Logical Conditions (AND, OR, NOT)

A logical condition combines the result of two component conditions to produce a single result based on them or inverts the result of a single condition. A row is returned only if the overall result of the condition is true.

Operator	**Meaning**
AND	Returns TRUE if *both* component conditions are true
OR	Returns TRUE if *either* component condition is true
NOT	Returns TRUE if the following condition is false

```
1. SELECT empno, ename, job, sal FROM emp WHERE sal >= 2000 AND job
   LIKE '%MAN%';
2. SELECT empno, ename, job, sal FROM emp WHERE sal >= 2000 OR job
   LIKE '%MAN%';
3. SELECT empno, ename, job, sal FROM emp WHERE job NOT IN
   ('PRESIDENT', 'CLERK');
```

22. ORDER BY Clause

The order of rows returned in a query result is undefined. The ORDER BY clause can be used to sort the rows. If you use the ORDER BY clause, it must be the last clause of the SELECT statement. The value **ASC** will be used to sort data in ascending order (default order), and **DESC** will sort the data in descending order.

```
1. SELECT ename, job, sal FROM emp ORDER BY hiredate;
2. SELECT ename, job, sal FROM emp ORDER BY empno DESC;
```

You can also sort using the column alias in the ORDER BY clause.

```
SELECT ename, sal, 12 * sal AS NewSal FROM emp ORDER
BY NewSal;
```

You can also sort query result on more than one column.

```
SELECT empno, ename, sal, deptno FROM emp ORDER BY deptno, sal DESC;
```

You can also sort by a column that is not included in the SELECT list.

The End