# Department of Computer Science

**420-231-VA Database**

Teacher: Mehdi Farshid Farzad

**Lab 01 – Simple SELECT Query**

**Instructions:**

1. **Please demonstrate** your work to your teacher, then submit your lab as a script file **Lab1\_script.sql**.
2. The lab must be submitted on Omnivox (Lea) before the end of the class.

# What to do:

1. Run Microsoft SQL Server Management Studio.
2. Open “New Query” editor, and create a database: create database Lab\_01;
3. Use this database for your operations: use Lab\_01;
4. Then, download the SQL Script from Omnivox. This SQL Script will create a Table “Student” in your database server. Then open the .sql file and Execute it:

File > Open > File… > Select the File > Execute

Or you may use the following script:

-- Create Student table

CREATE TABLE Student (

StudentID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Age INT,

Grade FLOAT,

Major VARCHAR(50)

);

-- Insert sample records

INSERT INTO Student VALUES (1, 'John', 'Doe', 20, 85.5, 'Computer Science');

INSERT INTO Student VALUES (2, 'Jane', 'Smith', 22, 78.0, 'Biology');

INSERT INTO Student VALUES (3, 'Alex', 'Johnson', 21, 92.3, 'Mathematics');

INSERT INTO Student VALUES (4, 'Emily', 'Williams', 19, 76.8, 'Physics');

INSERT INTO Student VALUES (5, 'Chris', 'Brown', 23, 88.7, 'Chemistry');

INSERT INTO Student VALUES (6, 'Megan', 'Taylor', 20, 94.2, 'Psychology');

INSERT INTO Student VALUES (7, 'Ryan', 'Miller', 22, 79.5, 'English');

INSERT INTO Student VALUES (8, 'Sophia', 'Anderson', 21, 87.1, 'History');

The given Student table consists of 6 COLUMNS and 8 ROWS. The name of the 6 COLUMNS are:

StudentID, FirstName, LastName, Age, Grade, Major

To perform any kind of activities with this table, we have to write and execute SQL queries to access the information from the given Table. For our lab, we will only play with the SELECT query, the easiest one. This query will help us view the data stored in the table and perform any mathematical extensive operation if required.

# SELECT Query:

The SELECT query specifies the columns to be retrieved as a column list. To list all the rows of the STUDENT table, write the following query:

**SELECT** StudentID, FirstName, LastName, Age, Grade, Major

**FROM STUDENT;**

**OR SELECT \* FROM STUDENT;**

If you want to access all the columns of a table, you can only put \* instead of giving the whole column list. Once you execute the query, you will see the following result set:



**Arithmetic Operations:**

Within SELECT statement you can create a query with number and column values using arithmetic operators. The arithmetic operators are +, -, \*, /

Let’s say, we want to add 2 marks to Age and deduct 1 mark from Grade for all the students in the class (use aliases to give new columns names). To do so, we need to perform the following query:

SELECT Age + 2 AS NEW\_Age, Grade -1 NEW\_Grade FROM STUDENT;

# Write the following queries:

1. As an instructor, you need to see your students’ ID, FirstName, LastName and Major altogether. Please write down a query that will help you to get the information that popped up from the given table.
2. As an instructor, you want to add 5% marks to your students’ Grade.

# 

# Where Clause:

You can select partial table contents by placing restrictions on the rows to be included in the output. Use the WHERE clause to add conditional restrictions to the SELECT statement that limit the rows returned by the query.

Let us perform the first query. We want to see all the information of the student whose “id” is

3:

**SELECT \* FROM STUDENT WHERE studentID = 3;**

Sometimes your search involves multiple conditions. For example, when you are buying a new house, you look for a certain area, a certain number of bedrooms, bathrooms, and so on. In the same way, SQL allows you to include multiple conditions in a query using logical operators. The logical operators are AND, OR, and NOT.

## **Comparison Operators**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| = | Equal to | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_equal_to) |
| > | Greater than | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_greater_than) |
| < | Less than | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_less_than) |
| >= | Greater than or equal to | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_greater_than2) |
| <= | Less than or equal to | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_less_than2) |
| <> | Not equal to |  |

## **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.

### Example: Select all students with a grade between 85 and 100

SELECT \* FROM STUDENT  
WHERE GRADE BETWEEN 85 AND 100;

## **NOT BETWEEN**

To display the students outside the range of the previous example, use NOT BETWEEN:

### Example:

### SELECT \* FROM STUDENT WHERE GRADE NOT BETWEEN 85 AND 100;

Or

SELECT \* FROM STUDENT

WHERE FIRSTNAME BETWEEN 'ALEX' AND 'MEGAN';

## **The SQL 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.

### Example:

Show all students with age of 20 or 22

SELECT \* FROM STUDENT

WHERE AGE = 20 or AGE = 22;

And now:

SELECT \* FROM STUDENT  
WHERE AGE IN (20 , 22);

# Order BY clause

Rows can be ordered based on the values of a column. You can view them in ascending or descending order. (ASC: Ascending Order, DESC: Descending Order)

Let’s say we are planning to see our students record in ascending order basis on their final marks.

SELECT \* FROM STUDENT ORDER BY GRADE ASC;

If we want to see descending order of their final exam, we can put the following query:

SELECT \* FROM STUDENT ORDER BY GRADE DESC;

## **SELECT TOP Clause**

The SELECT TOP clause is used to specify the number of records to return.

The SELECT TOP clause is useful on large tables with thousands of records. Returning a large number of records can impact performance.

### **Example**:

Select only the first 3 records of the student table:

SELECT TOP 3 \* FROM STUDENT;

# Write the following queries:

1. Select all columns for all students.
2. Select only the first and last names of students.
3. Select the student with StudentID 3.
4. Select students with an age greater than 21.
5. Select students majoring in 'Chemistry'.
6. Select students with a grade higher than 80.
7. Select students majoring in 'Physics' and with a grade below 80.
8. Select students aged 20 or 21.
9. Select students with a grade between 85 and 90.
10. Select students majoring in 'Computer Science' or 'Mathematics'.
11. Select all columns and order the result by age in descending order.
12. Select the top 3 students with the highest grades.
13. Select the top 2 oldest students.
14. Select the student with the lowest grade.