IT Department

Structured Query Language (SQL)

Soma Soleiman Zadeh

Database Systems I (IT 215)

Fall 2025 - 2026

Week 8

November 24, 2025

Outline

- What is **Query**?
- Procedural Language vs. Non-Procedural Language
- Structured Query Language (SQL)
- **DDL** and **DML** Statements
- Basic SQL **SELECT** Query Clauses

What is Query?

- A **database query** is a request that is sent to a database management system (DBMS) in order to retrieve or manipulate data in database.
- Query can be used to:
  - Create and delete databases
  - Create and delete tables
  - Insert records to a database
  - Retrieve data from a database
  - Update records in a database
  - Delete records from a database.

SQL (Structured Query Language)

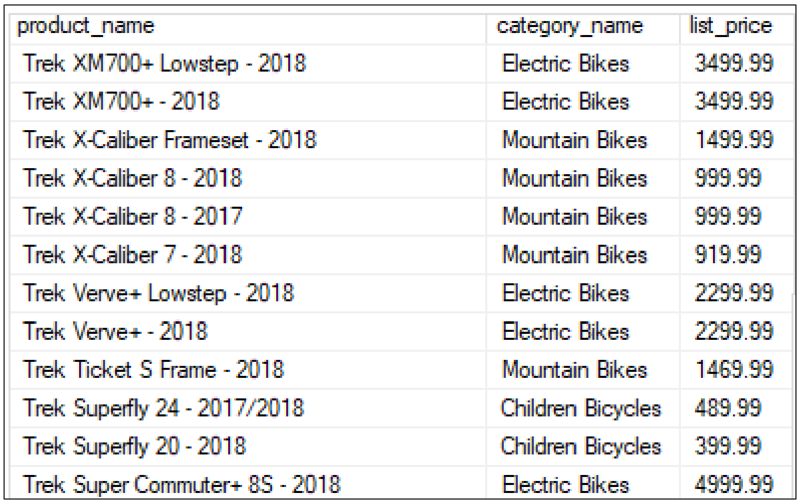
- **SQL** is a database language that allows a user to:
  - **create the database and table structures**,
  - **perform basic data management tasks**, such as the insertion, modification, and deletion of data from the relations,
  - **perform both simple and complex queries**.
- **MS Access** and **MySQL** use SQL.

Procedural vs. Non-Procedural Language

- **Procedural** Language:
  - User instructs the system to perform a sequence of operations to get the desired information.
  - **Examples**: Java, C, C#
- **Non-procedural** Language:
  - User specifies what information they require, <u>without describing how to get it</u>.
  - **Example**: SQL

Difference between Procedural and Non-Procedural Languages

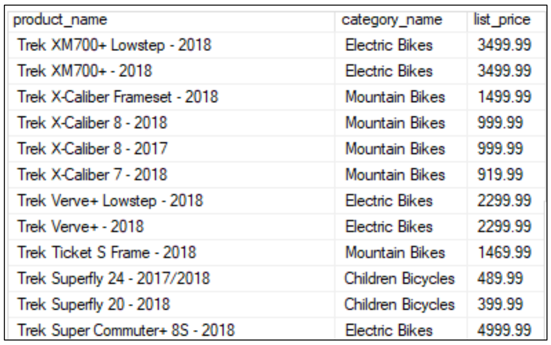
Suppose we have a **Product** table and <u>we want to know which products have a price of more than $1000.</u>



product_name	category_name	list_price
Trek XM700+ Lowstep - 2018	Electric Bikes	3499.99
Trek XM700+ - 2018	Electric Bikes	3499.99
Trek X-Caliber Frameset - 2018	Mountain Bikes	1499.99
Trek X-Caliber 8 - 2018	Mountain Bikes	999.99
Trek X-Caliber 8 - 2017	Mountain Bikes	999.99
Trek X-Caliber 7 - 2018	Mountain Bikes	919.99
Trek Verve+ Lowstep - 2018	Electric Bikes	2299.99
Trek Verve+ - 2018	Electric Bikes	2299.99
Trek Ticket S Frame - 2018	Mountain Bikes	1469.99
Trek Superfly 24 - 2017/2018	Children Bicycles	489.99
Trek Superfly 20 - 2018	Children Bicycles	399.99
Trek Super Commuter+ 8S - 2018	Electric Bikes	4999.99

Procedural Language

```
for each product in products
        if product_price > 1000
                add to expensive_product_array
        else
                ignore
        end
end
return expensive_product_array
```



product_name	category_name	list_price
Trek XM700+ Lowstep - 2018	Electric Bikes	3499.99
Trek XM700+ - 2018	Electric Bikes	3499.99
Trek X-Caliber Frameset - 2018	Mountain Bikes	1499.99
Trek X-Caliber 8 - 2018	Mountain Bikes	999.99
Trek X-Caliber 8 - 2017	Mountain Bikes	999.99
Trek X-Caliber 7 - 2018	Mountain Bikes	919.99
Trek Verve+ Lowstep - 2018	Electric Bikes	2299.99
Trek Verve+ - 2018	Electric Bikes	2299.99
Trek Ticket S Frame - 2018	Mountain Bikes	1469.99
Trek Superfly 24 - 2017/2018	Children Bicycles	489.99
Trek Superfly 20 - 2018	Children Bicycles	399.99
Trek Super Commuter+ 8S - 2018	Electric Bikes	4999.99

Non-Procedural Language (SQL)

○ We describe what we want:

❑ **In English:**

"I want all products with a price of more than $ 1000"

❑ **In SQL:**

```
SELECT * FROM products WHERE price > 1000;
```

SQL Language Major Components

- The **SQL** standard has two major components:
  - **Data Definition Language (DDL)** for
    - ✓ **defining** the database structure (table schemas), and
    - ✓ **deleting** tables and **modifying** table schemas.
  - **Data Manipulation Language (DML)** for
    - ✓ **retrieving (requesting) the data** from tables, and
    - ✓ **updating, deleting, and inserting data** in tables.

DDL or DML Statements?

```
CREATE  TABLE  Book
(BID   varchar(3),
Name  varchar(40),
Author  varchar(30),
PRIMARY  KEY  (BID) )
```

```
INSERT INTO Book VALUES (12, 'Intro to Java' , 'Mike');
```

```
DELETE FROM Book;
```

```
ALTER TABLE Book ADD pages int;
```

SQL DML – Manipulating the Database

◦ The SQL **DML** statements are:

SELECT	To query (request) data in the database
UPDATE	To update data in a table
INSERT	To insert data in a table
DELETE	To delete data from a table

SELECT Statement

◦ **SELECT** statement is used by users and applications to get and display their desired data from one or more database tables.

◦ **SELECT** is the most frequently used SQL command.

◦ **SQL SELECT** command syntax:

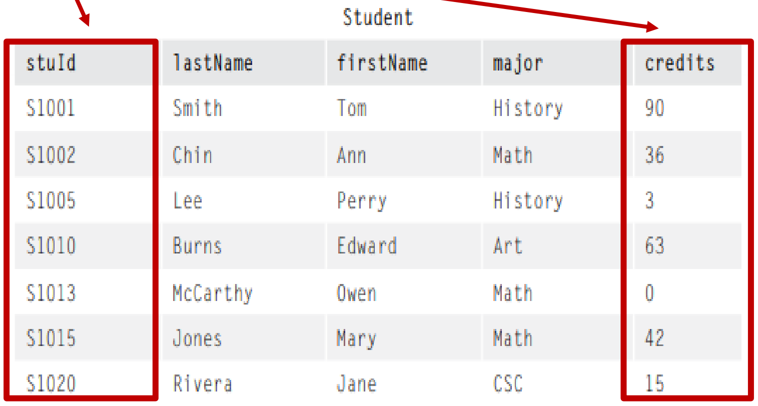
```
SELECT   desired_column(s)
FROM     table(s)
WHERE    condition(s)
```

◦ The result of an SQL query is a **table**.

SELECT statement (Simple Example)

○ Write an SQL query to get **ID** and **credits** of all students.

```
SELECT    stuId , credits
FROM      Student;
```



Student

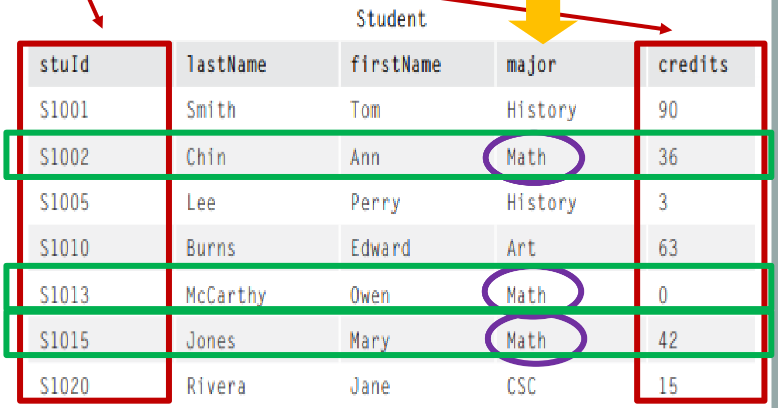
stuId	lastName	firstName	major	credits
S1001	Smith	Tom	History	90
S1002	Chin	Ann	Math	36
S1005	Lee	Perry	History	3
S1010	Burns	Edward	Art	63
S1013	McCarthy	Owen	Math	0
S1015	Jones	Mary	Math	42
S1020	Rivera	Jane	CSC	15

SELECT statement (Simple Example)

○ Write an SQL query to get **ID** and **credits** of those students that their **major** is 'Math'.

```
SELECT    stuId , credits
FROM      Student
WHERE     major = 'Math';
```

stuId	credits
S1002	36
S1013	0
S1015	42



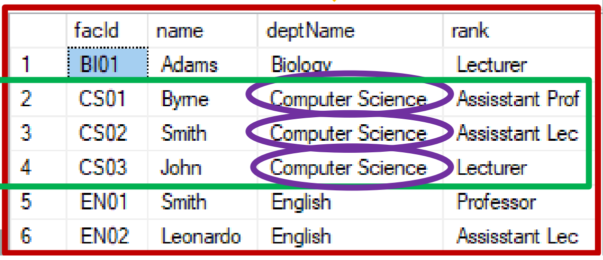
Student

stuId	lastName	firstName	major	credits
S1001	Smith	Tom	History	90
S1002	Chin	Ann	Math	36
S1005	Lee	Perry	History	3
S1010	Burns	Edward	Art	63
S1013	McCarthy	Owen	Math	0
S1015	Jones	Mary	Math	42
S1020	Rivera	Jane	CSC	15

SELECT statement (Use of *)

- Write an SQL query to get <u>all information</u> of faculties in **'Computer Science' department**.

```
SELECT    *
FROM      Faculty
WHERE     deptName = 'Computer Science';
```



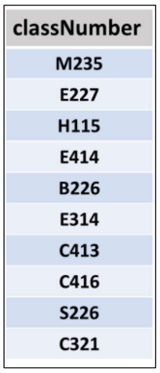
	facId	name	deptName	rank
1	BI01	Adams	Biology	Lecturer
2	CS01	Byrne	Computer Science	Assistant Prof
3	CS02	Smith	Computer Science	Assistant Lec
4	CS03	John	Computer Science	Lecturer
5	EN01	Smith	English	Professor
6	EN02	Leonardo	English	Assistant Lec

SELECT statement (Using DISTINCT)

- Write an SQL query to get **class number** of all classes in which students are enrolled.

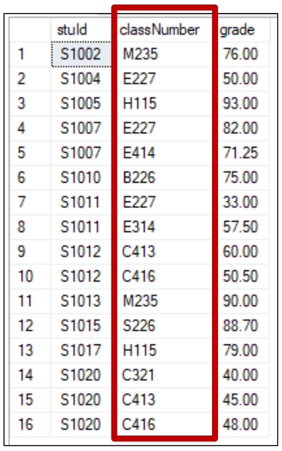
  (With removing duplicated values)

```
SELECT DISTINCT  classNumber
FROM      Enroll;
```



classNumber
M235
E227
H115
E414
B226
E314
C413
C416
S226
C321

Enroll Table



	stuId	classNumber	grade
1	S1002	M235	76.00
2	S1004	E227	50.00
3	S1005	H115	93.00
4	S1007	E227	82.00
5	S1007	E414	71.25
6	S1010	B226	75.00
7	S1011	E227	33.00
8	S1011	E314	57.50
9	S1012	C413	60.00
10	S1012	C416	50.50
11	S1013	M235	90.00
12	S1015	S226	88.70
13	S1017	H115	79.00
14	S1020	C321	40.00
15	S1020	C413	45.00
16	S1020	C416	48.00

SELECT Statement (Arithmetic Operations)

◦ Arithmetic operations can be written within select clause to perform calculations.

```
SELECT    employeeName, Salary/2
FROM      Employee;
```

Employee

employeeID	employeeName	Salary
65	Kate	2000
77	Mike	4000
80	John	1000

Output

employeeName	Salary
Kate	1000
Mike	2000
John	500

The Where Clause (Use of Multiple Conditions)

◦ The SQL **Where** clause specifies condition(s) that the result must satisfy.

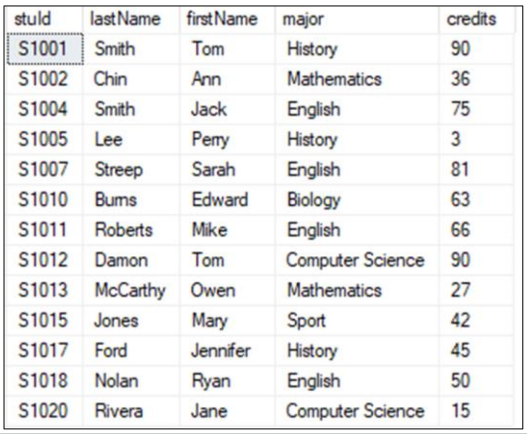
◦ Comparison results can be combined using the logical connectives **and**, **or**, and **not**.

The Where Clause (Use of Multiple Conditions)

- Write an SQL query to find **first name** and **last name** of all students whose **major** is '**English**' and have **more than 70 credits.**

```
SELECT    lastName, firstName
FROM      Student
WHERE     major = 'English' AND credits > 70;
```

Student



stuId	lastName	firstName	major	credits
S1001	Smith	Tom	History	90
S1002	Chin	Ann	Mathematics	36
S1004	Smith	Jack	English	75
S1005	Lee	Perry	History	3
S1007	Streep	Sarah	English	81
S1010	Burns	Edward	Biology	63
S1011	Roberts	Mike	English	66
S1012	Damon	Tom	Computer Science	90
S1013	McCarthy	Owen	Mathematics	27
S1015	Jones	Mary	Sport	42
S1017	Ford	Jennifer	History	45
S1018	Nolan	Ryan	English	50
S1020	Rivera	Jane	Computer Science	15

Example

- Find **name** and **age** of students who study in **IT** or **Cybersecurity** departments.

Student

stuID	stuName	Year	age	deptName
16	Mike	2nd	23	IT
17	Peter	3rd	22	Comp. Eng.
18	Kelly	3rd	31	IT
19	Roberto	2nd	19	Comp. Eng.
20	Sara	2nd	25	Cybersecurity

```
SELECT    firstName, lastName, age
FROM      Student
WHERE     major = 'IT' OR major = 'Cybersecurity' ;
```

```
SELECT    firstName, lastName, age
FROM      Student
WHERE     major IN ('IT' , 'Cybersecurity' );
```

Example

- Find **name** and **age** of **2nd year students** who study in **IT** or **Cybersecurity** departments.

Student

stuID	stuName	Year	age	deptName
16	Mike	2nd	23	IT
17	Peter	3rd	22	Comp. Eng.
18	Kelly	3rd	31	IT
19	Roberto	2nd	19	Comp. Eng.
20	Sara	2nd	25	Cybersecurity

```
SELECT   firstName, lastName, age
FROM     Student
WHERE    major IN (‘IT’ , ‘Cybersecurity’ ) AND Year = ‘2nd’;
```

Example

- Find all information of students whose **age** are between 20 and 30.

Student

stuID	stuName	Year	age	deptName
16	Mike	2nd	23	IT
17	Peter	3rd	22	Comp. Eng.
18	Kelly	3rd	31	IT
19	Roberto	2nd	19	Comp. Eng.
20	Sara	2nd	25	Cybersecurity

```
SELECT   *
FROM     Student
WHERE    age >= 20  AND  age <= 30;
```

```
SELECT   *
FROM     Student
WHERE    age  BETWEEN 20  AND 30;
```

The Where Clause (Cont.)

- The predicate **is null** is used to check for null values.
- **Example:** Find name of students whose age are not written (is null).

Student

stuID	stuName	Year	age	deptName
16	Mike	2nd	23	IT
17	Peter	3rd	22	Comp. Eng.
18	Kelly	3rd	31	IT
19	Roberto	2nd		Comp. Eng.
20	Sara	2nd	25	Cybersecurity

```
SELECT   stuName
FROM     Student
WHERE    age  IS  NULL;
```

SELECT statement (Use of ORDER BY)

- The **ORDER BY** option in SQL SELECT allows us to order the retrieved records in ascending (**ASC**—the default) or descending (**DESC**) order on any field or combination of fields.

	Ascending Order (ASC Keyword)	Descending Order (DESC Keyword)
Numbers	From Lowest to Highest	From Highest to Lowest
Characters	From A to Z	From Z to A

ORDER BY Example

- Write a Query to find **name** and **ID** of all teachers. The result is arranged in alphabetic order of teachers' names.

Teacher

teacherID	teacherName	deptName	teacherRank
BI01	Adams	Biology	Lecturer
CS01	Byrne	Computer Science	Assistant Prof
CS02	Smith	Computer Science	Assistant Lec
CS03	John	Computer Science	Lecturer
EN01	Smith	English	Professor
EN02	Leonardo	English	Assistant Lec
EN03	Kate	English	Lecturer
HI01	Kim	History	Assistant Prof
MA01	Julia	Mathematics	Assistant Lec
SP01	Maria	Sport	Professor
SP02	Sarah	Sport	Lecturer

```
SELECT      teacherName, teacherID
FROM        Teacher
ORDER BY    teacherName;
```

teacherName	teacherID
Adams	BI01
Byrne	CS01
John	CS03
Julia	MA01
Kate	EN03
Kim	HI01
Leonardo	EN02
Maria	SP01
Sarah	SP02
Smith	CS02
Smith	EN01

ORDER BY Example

- Write a Query to find **name** and **ID** of all teachers. The result is arranged in alphabetic order of teachers' names.
- For teachers with the same name, order them in alphabetic order of their **department names.**

Teacher

teacherID	teacherName	deptName	teacherRank
BI01	Adams	Biology	Lecturer
CS01	Byrne	Computer Science	Assistant Prof
CS02	Smith	Computer Science	Assistant Lec
CS03	John	Computer Science	Lecturer
EN01	Smith	English	Professor
EN02	Leonardo	English	Assistant Lec
EN03	Kate	English	Lecturer
HI01	Kim	History	Assistant Prof
MA01	Julia	Mathematics	Assistant Lec
SP01	Maria	Sport	Professor
SP02	Sarah	Sport	Lecturer

```
SELECT      teacherName, teacherID
FROM        Teacher
ORDER BY    teacherName, deptName;
```

teacherName	teacherID
Adams	BI01
Byrne	CS01
John	CS03
Julia	MA01
Kate	EN03
Kim	HI01
Leonardo	EN02
Maria	SP01
Sarah	SP02
Smith	CS02
Smith	EN01