Tishk International University
Department of Information Technology
Database Systems 1
Week 9
Fall 2023-24
November 26, 2023

# Some Operators and Clauses

Wisam Abdulaziz Qadir
Wisam.abdulaziz@tiu.edu.iq

# Outline


- SQL IN Operator
- SQL NOT EQUAL Operator
- BETWEEN Operator
- SELECT TOP Clause
- Having Clause

# IN Operator


- The IN operator allows us to specify multiple values in a WHERE clause.

- The IN operator is shorthand for multiple **OR** conditions.

Syntax:

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

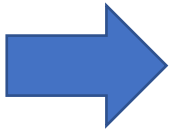
# IN Operator (cont.)

- E.g.: Retrieve full name and department of all students who study IT, Arch and Civil.

```
SELECT  f_name, l_name, dept
FROM    student
WHERE dept IN ('IT', 'Arch', 'Civil');
```

| SID | F_name | L_name | Dept |
|---|---|---|---|
| 1 | Dara | Azad | IT |
| 2 | Zara | Nawzad | Biology |
| 3 | Ali | Omer | Arch |
| 4 | Nasrin | Dana | IT |
| 5 | Aras | Zana | Civil |

**Student**



| F_name | L_name | Dept |
|---|---|---|
| Dara | Azad | IT |
| Ali | Omer | Arch |
| Nasrin | Dana | IT |
| Aras | Zana | Civil |

# IN Operator (cont.)

- E.g.: Retrieve full name and department of all students who don't study IT and Arch.

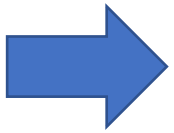
```
SELECT  f_name, l_name, dept
FROM    student
WHERE dept NOT IN ('IT', 'Arch');
```

=

```
SELECT  f_name, l_name, dept
FROM    student
WHERE  NOT dept = 'IT' OR NOT dept = 'Arch';
```

| SID | F_name | L_name | Dept |
|---|---|---|---|
| 1 | Dara | Azad | IT |
| 2 | Zara | Nawzad | Biology |
| 3 | Ali | Omer | Arch |
| 4 | Nasrin | Dana | IT |
| 5 | Aras | Zana | Civil |

**Student**



| F_name | L_name | Dept |
|---|---|---|
| Zara | Nawzad | Biology |
| Aras | Zana | Civil |

# Not Equal Operator

- Checks if values of two operands are equal or not, if values are **not equal** then condition becomes true.

- The **Not Equal** operator is written as ( **<>** ) or ( **!=** ) in SQL.

Syntax:

```
SELECT  column_name(s)
FROM    table_name
WHERE column_name <> value;
```

```
SELECT  column_name(s)
FROM    table_name
WHERE column_name != value;
```

# Not Equal Operator (cont.)

- E.g.: Retrieve full name and department of all students who don't study IT.

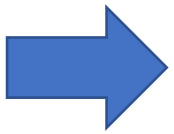
```
SELECT  f_name, l_name, dept
FROM    student
WHERE dept <> 'IT';
```

=

```
SELECT  f_name, l_name, dept
FROM    student
WHERE dept != 'IT';
```

| SID | F_name | L_name | Dept |
|---|---|---|---|
| 1 | Dara | Azad | IT |
| 2 | Zara | Nawzad | Biology |
| 3 | Ali | Omer | Arch |
| 4 | Nasrin | Dana | IT |
| 5 | Aras | Zana | Civil |

**Student**



| F_name | L_name | Dept |
|---|---|---|
| Zara | Nawzad | Biology |
| Ali | Omer | Arch |
| Aras | Zana | Civil |

# BETWEEN Operator

- It selects values within a given range.

Syntax:

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

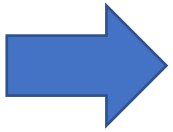
# BETWEEN Operator (cont.)

- E.g.: Retrieve students whose mark is greater than 79 and less than 95.

```
SELECT  *
FROM   Student
WHERE Mark BETWEEN 79 AND 95;
```

| SID | F_name | L_name | Mark |
|---|---|---|---|
| 1 | Dara | Azad | 75 |
| 2 | Zara | Nawzad | 90 |
| 3 | Ali | Omer | 80 |
| 4 | Nasrin | Dana | 100 |
| 5 | Aras | Zana | 78 |

**Student**



| SID | F_name | L_name | Mark |
|---|---|---|---|
| 2 | Zara | Nawzad | 90 |
| 3 | Ali | Omer | 80 |

# SELECT TOP Clause

- Is used to specify the number of records to return.

Syntax:

```
SELECT TOP number column_name(s)
FROM table_name
WHERE condition;
```

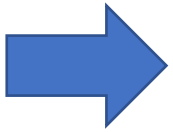
# SELECT TOP Clause (cont.)

- E.g.: Retrieve information of only the first 2 students in the table.

```
SELECT  TOP 2 *
FROM    Student;
```

| SID | F_name | L_name | Mark |
|---|---|---|---|
| 1 | Dara | Azad | 75 |
| 2 | Zara | Nawzad | 90 |
| 3 | Ali | Omer | 80 |
| 4 | Nasrin | Dana | 100 |
| 5 | Aras | Zana | 78 |

**Student**



| SID | F_name | L_name | Mark |
|---|---|---|---|
| 1 | Dara | Azad | 75 |
| 2 | Zara | Nawzad | 90 |

# SELECT TOP Clause (cont.)

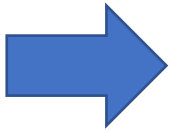
- E.g.: Retrieve information of only top 2 students (highest 2 marks).

Step 1:

```
SELECT  *
FROM   Student
ORDER BY Mark DESC;
```

| SID | F_name | L_name | Mark |
|---|---|---|---|
| 1 | Dara | Azad | 75 |
| 2 | Zara | Nawzad | 90 |
| 3 | Ali | Omer | 80 |
| 4 | Nasrin | Dana | 100 |
| 5 | Aras | Zana | 78 |

**Student**



| SID | F_name | L_name | Mark |
|---|---|---|---|
| 4 | Nasrin | Dana | 100 |
| 2 | Zara | Nawzad | 90 |
| 3 | Ali | Omer | 80 |
| 5 | Aras | Zana | 78 |
| 1 | Dara | Azad | 75 |

**Student_Order_Query**

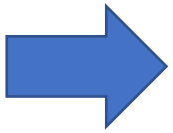
# SELECT TOP Clause (cont.)

Step 2:

```
SELECT  TOP 2 *
FROM   Student_Order_Query;
```

| SID | F_name | L_name | Mark |
|---|---|---|---|
| 4 | Nasrin | Dana | 100 |
| 2 | Zara | Nawzad | 90 |
| 3 | Ali | Omer | 80 |
| 5 | Aras | Zana | 78 |
| 1 | Dara | Azad | 75 |

**Student_Order_Query**



| SID | F_name | L_name | Mark |
|---|---|---|---|
| 4 | Nasrin | Dana | 100 |
| 2 | Zara | Nawzad | 90 |

**Top_2_Students**

# HAVING Clause

- The **HAVING** clause was added to SQL because aggregate functions cannot be used within the **WHERE** clause.

Syntax:

**Without** ORDER BY

**SELECT** *column_name(s)*
**FROM** *table_name*
**GROUP BY** *column_name(s)*
**HAVING** *condition*

**With** ORDER BY

**SELECT** *column_name(s)*
**FROM** *table_name*
**GROUP BY** *column_name(s)*
**HAVING** *condition*
**ORDER BY** *column_name(s)*

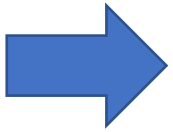
# HAVING Clause (cont.)

- E.g.: Retrieve only those department names that have more than 3 students.

```
SELECT COUNT(SID) as Student_no, Dept
FROM Student
GROUP BY Dept
HAVING COUNT(SID) > 3;
```

| SID | F_name | L_name | Dept |
|---|---|---|---|
| 1 | Dara | Azad | IT |
| 2 | Zara | Nawzad | Civil |
| 3 | Ali | Omer | IT |
| 4 | Nasrin | Dana | Civil |
| 5 | Aras | Zana | IT |
| 6 | Kawa | Kamaran | IT |

**Student**



| Student_no | Dept_no |
|---|---|
| 4 | IT |

# Thank You