SELECT

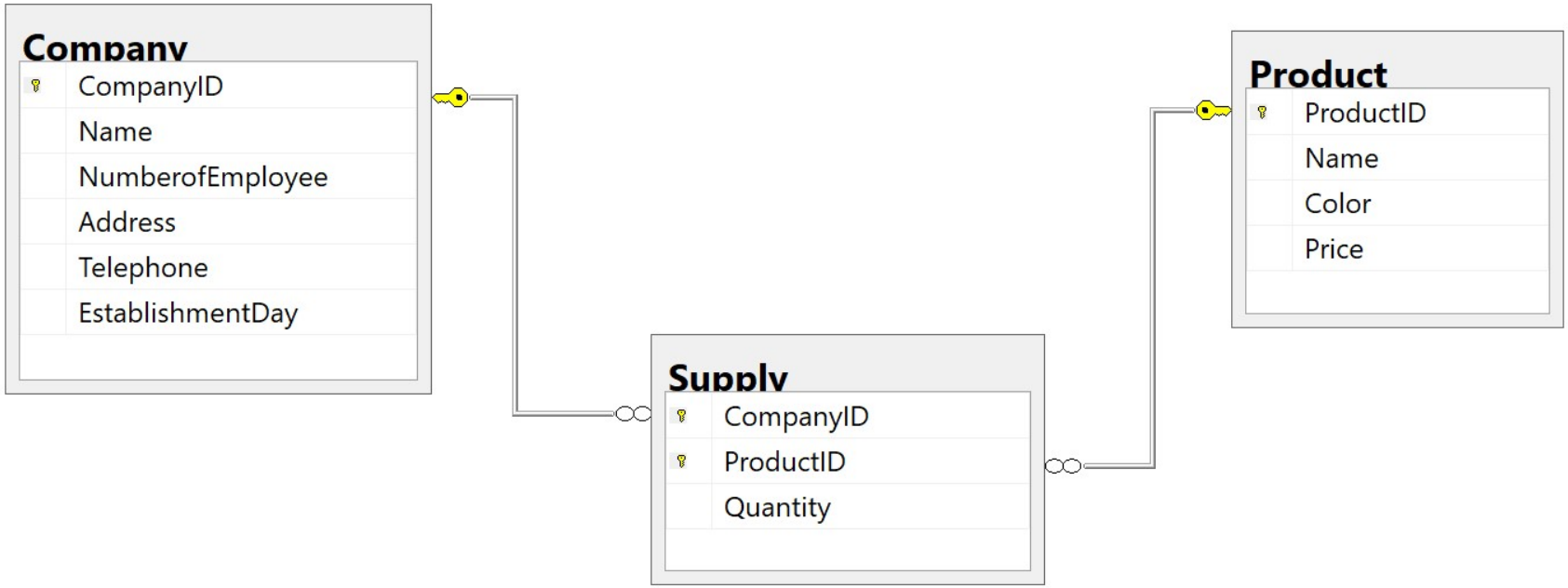
NGUYEN Hong Phuong
Email: phuongnh@soict.hust.edu.vn
Site: https://users.soict.hust.edu.vn/phuongnh

Contents

- Database sample
- SELECT Statement
- DISTINCT
- Comments
- WHERE Clause
- AND, OR and NOT Operators
- ORDER BY
- SELECT TOP Clause
- LIKE Operator
- Wildcard Characters
- IN Operator
- BETWEEN Operator
- Aliases
- Joins
- UNION Operator
- INTERSECT Operator
- EXCEPT Operator
- GROUP BY Statement
- EXISTS Operator
- ANY and ALL Operators
- CASE Statement
- SQL Operators
- Functions

Database sample

- Company-Supply-Product
- Database diagram



SELECT Statement

- The SELECT statement is used to select data from a database.
- The data returned is stored in a result table, called the result-set.
- SELECT Syntax

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

  - Column1, column2, ... are the field names of the table you want to select data from.
- To select all the fields available in the table, use the following syntax:

```
SELECT * FROM table_name;
```

SELECT Column Example

- The following SQL statement selects the "Name", "NumberofEmployee" and "Address" columns from the "Company" table:

```
SELECT Name, NumberofEmployee, Address
FROM Company
```

- The following SQL statement selects all the columns from the "Company" table:

```
SELECT *
FROM Company
```

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.
- Syntax

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

SELECT DISTINCT (cont'd)

- SELECT Example Without DISTINCT

```
SELECT Address
FROM Company
```

Results | Messages

	Address
1	Seoul, Korea
2	LongBien, Hanoi
3	Michigan, US
4	Ingolstadt, Germany
5	Michigan, US
6	Maranello, Italy
7	Hiroshima, Japan
8	Aichi, Japan
9	Tokyo, Japan
10	Munchen, Germany
11	Coventry, UK
12	London, UK
13	London, UK
14	Baden-Wurttemberg, Germany
15	Baden-Wurttemberg, Germany
16	Paris, France
17	Tokyo, Japan

- SELECT DISTINCT Examples

```
SELECT DISTINCT Address
FROM Company
```

Results | Messages

	Address
1	Aichi, Japan
2	Baden-Wurttemberg, Germany
3	Coventry, UK
4	Hiroshima, Japan
5	Ingolstadt, Germany
6	London, UK
7	LongBien, Hanoi
8	Maranello, Italy
9	Michigan, US
10	Munchen, Germany
11	Paris, France
12	Seoul, Korea
13	Tokyo, Japan

Comments

- Comments are used to explain sections of SQL statements, or to prevent execution of SQL statements.
- Single Line Comments
  - Single line comments start with --.
  - Any text between -- and the end of the line will be ignored (will not be executed).
  - The following example uses a single-line comment as an explanation:

```
--Select all:
SELECT * FROM Company
```

SQL WHERE Clause

- Is used to filter records.
- Is used to extract only those records that fulfill a specified condition.
- Syntax

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

- Example

```
SELECT *
FROM Company
WHERE Address = 'Tokyo, Japan'
```

```
SELECT *
FROM Company-- WHERE Address = 'Tokyo, Japan'
```

SQL WHERE Clause (cont'd)

- Text Fields vs. Numeric Fields
  - SQL requires single quotes around text values (most database systems will also allow double quotes).
  - However, numeric fields should not be enclosed in quotes:

```
SELECT *
FROM Company
WHERE CompanyID = 12
```

SQL WHERE Clause (cont'd)

- Operators in The WHERE Clause

Operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=
BETWEEN	Between a certain range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

SQL AND, OR and NOT Operators

- 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.

SQL AND, OR and NOT Operators (cont'd)

AND Syntax

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

OR Syntax

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

NOT Syntax

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

SQL AND, OR and NOT Operators (cont'd)

- Examples

```
SELECT *
FROM Company
WHERE Address = 'London, UK' AND NumberofEmployee>3500
```

```
SELECT *
FROM Company
WHERE Address = 'London, UK' OR Address = 'Michigan, US'
```

```
SELECT * FROM Company
WHERE NOT Address = 'London, UK'
```

SQL AND, OR and NOT Operators (cont'd)

- Combining AND, OR and NOT

```
SELECT *
FROM Company
WHERE NumberofEmployee > 3500
AND (Address = 'Tokyo, Japan' OR Address = 'London, UK')
```

```
SELECT *
FROM Company
WHERE NOT Address = 'Tokyo, Japan' AND NOT Address = 'London, UK'
```

ORDER BY

- Is used to sort the result-set in ascending or descending order.
- Sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.
- Syntax

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

- Example

```
SELECT *
FROM Company
ORDER BY Name ASC
```

```
SELECT *
FROM Company
ORDER BY Name DESC
```

ORDER BY (cont'd)

- ORDER BY Several Columns

```
SELECT *
FROM Company
ORDER BY Address, NumberofEmployee
```

```
SELECT *
FROM Company
ORDER BY Address ASC, NumberofEmployee DESC
```

SELECT TOP Clause

- is used to specify the number of records to return.
- is useful on large tables with thousands of records. Returning a large number of records can impact performance.
- Syntax

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

SELECT TOP Clause (cont'd)

- Example

```
SELECT TOP 3 * FROM Company
```

```
SELECT TOP 50 PERCENT * FROM Company
```

LIKE Operator

- is used in a WHERE clause to search for a specified pattern in a column.
- There are two wildcards often used in conjunction with the LIKE operator:
  - The percent sign (%) represents zero, one, or multiple characters
  - The underscore sign (_) represents one, single character
  - Notice: MS Access uses an asterisk (*) instead of the percent sign (%), and a question mark (?) instead of the underscore (_).

LIKE Operator (cont'd)

- Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern;
```

LIKE Operator	Description
WHERE Name LIKE 'a%'	Finds any values that start with "a"
WHERE Name LIKE '%a'	Finds any values that end with "a"
WHERE Name LIKE '%or%'	Finds any values that have "or" in any position
WHERE Name LIKE '_r%'	Finds any values that have "r" in the second position
WHERE Name LIKE 'a_%'	Finds any values that start with "a" and are at least 2 characters in length
WHERE Name LIKE 'a__%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE Name LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

LIKE Operator (cont'd)

- Example

```
SELECT * FROM Company
WHERE Name LIKE 'F%'

SELECT * FROM Company
WHERE Name LIKE '%a'

SELECT * FROM Company
WHERE Name LIKE '___'

SELECT * FROM Company
WHERE Name LIKE '%o_'
```

Wildcard Characters

- A wildcard character is used to substitute one or more characters in a string.
- Wildcard characters are used with the LIKE operator.

Wildcard Characters (cont'd)

- Wildcard Characters in SQL Server

Symbol	Description	Example
%	Represents zero or more characters	bl% finds bl, black, blue, and blob
_	Represents a single character	h_t finds hot, hat, and hit
[]	Represents any single character within the brackets	h[oa]t finds hot and hat, but not hit
^	Represents any character not in the brackets	h[^oa]t finds hit, but not hot and hat
-	Represents any single character within the specified range	c[a-b]t finds cat and cbt

Wildcard Characters (cont'd)

- Example

```
SELECT * FROM Company
WHERE Name LIKE '[HL]%'

SELECT * FROM Company
WHERE Name LIKE '[A-F]%'

SELECT * FROM Company
WHERE Name LIKE '[^A-F]%'
```

IN Operator

- allows you to specify multiple values in a WHERE clause.
- is a shorthand for multiple OR conditions.
- Syntax

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

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (SELECT STATEMENT);
```

IN Operator (cont'd)

- Example

```
SELECT * FROM Product
WHERE Color IN('red','blue','white')

SELECT * FROM Product
WHERE Color NOT IN('red','blue','white')

SELECT * FROM Company
WHERE CompanyID IN(SELECT CompanyID FROM Supply)

SELECT * FROM Company
WHERE CompanyID NOT IN(SELECT CompanyID FROM Supply)
```

BETWEEN Operator

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

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

BETWEEN Operator (cont'd)

- Example

```
SELECT * FROM Product
WHERE Price BETWEEN 4000 AND 12000;
```

```
SELECT * FROM Company
WHERE EstablishmentDay BETWEEN '1940/01/01' AND '2000/01/01'
```

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 that query.
- An alias is created with the AS keyword.
- Syntax
  - Column

```
SELECT column_name AS alias_name
FROM table_name;
```

  - Table

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

Aliases (cont'd)

- Example

```
SELECT CompanyID AS ID, Name AS Title
FROM Company

SELECT Name AS [Tên công ty], NumberofEmployee AS [Số nhân viên]
FROM Company

SELECT Name AS 'Tên công ty', NumberofEmployee AS 'Số nhân viên'
FROM Company

SELECT c.*
FROM Company AS c
WHERE c.Address LIKE '%Japan%'
```

Joins

- A JOIN clause is used to combine rows from two or more tables, based on a related column between them.
- Syntax

```
SELECT Column1, Column2, Column3,...
FROM table1, table2, table3,...
WHERE table1.Comlumn1 = table2.Column2
AND table2.Column3 = table3.Column4...
```

```
SELECT Column1, Column2, Column3,...
FROM table1 JOIN table2 ON table1.Comlumn1 = table2.Column2
JOIN table3 ON table2.Column3 = table3.Column4...
```

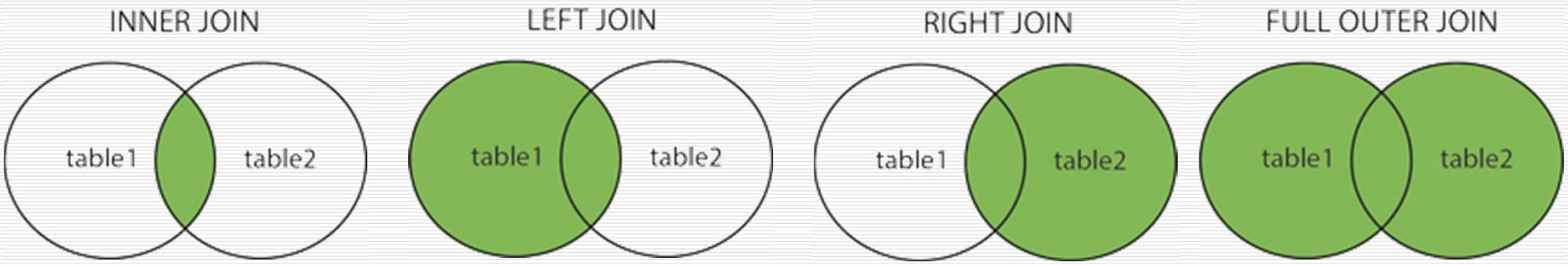
Joins (cont'd)

- Example

```
SELECT c.CompanyID, Name, ProductID
FROM Company c, Supply s
WHERE c.CompanyID = s.CompanyID
```

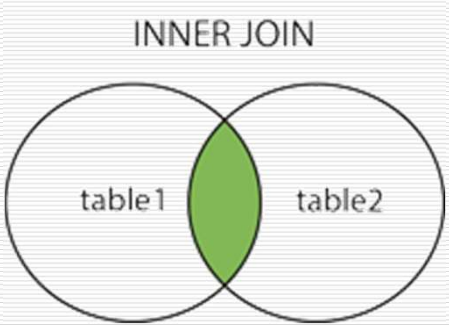
Joins (cont'd)

- Different Types of SQL JOINs
  - (INNER) JOIN: Returns records that have matching values in both tables
  - LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table
  - RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
  - FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table



Joins (cont'd)

- INNER JOIN
  - The INNER JOIN keyword selects records that have matching values in both tables.
  - Syntax




```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```

Joins (cont'd)

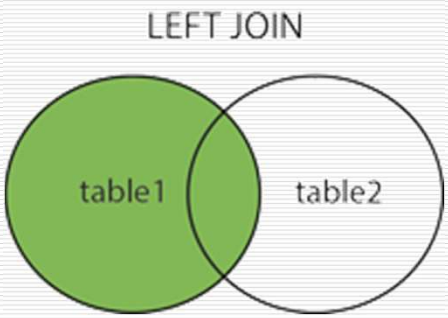
- INNER JOIN
  - Example

```
SELECT Company.CompanyID, Name, ProductID
FROM Company INNER JOIN Supply
ON Company.CompanyID = Supply.CompanyID
```

```
SELECT Company.Name, Product.Name, Quantity
FROM Company INNER JOIN Supply
ON Company.CompanyID = Supply.CompanyID
INNER JOIN Product
ON Supply.ProductID = Product.ProductID
```

Joins (cont'd)

- LEFT JOIN
  - returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match.
  - Syntax




```
SELECT column_name(s)
FROM table1 LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

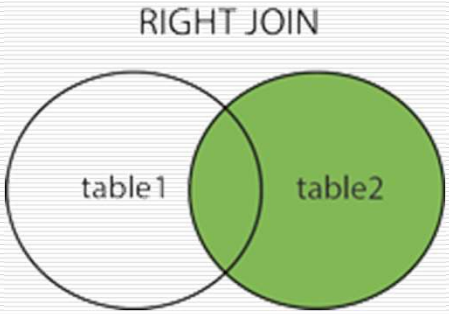
Joins (cont'd)

- LEFT JOIN
  - Example

```
SELECT Company.Name, Supply.ProductID
FROM Company LEFT JOIN Supply
ON Company.CompanyID = Supply.CompanyID
```

Joins (cont'd)

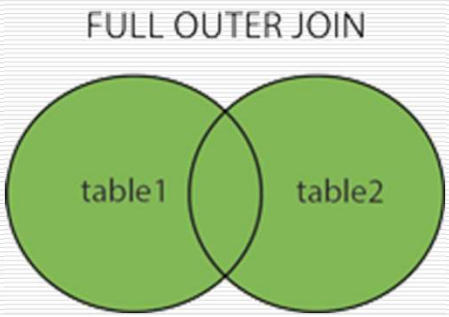
- RIGHT JOIN
  - returns all records from the right table (table2), and the matching records from the left table (table1). The result is 0 records from the left side, if there is no match.
  - Syntax




```
SELECT column_name(s)
FROM table1 RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

Joins (cont'd)

- FULL OUTER JOIN
  - returns all records when there is a match in left (table1) or right (table2) table records.
  - FULL OUTER JOIN and FULL JOIN are the same.
  - Syntax




```
SELECT column_name(s)
FROM table1 FULL OUTER JOIN table2
ON table1.column_name = table2.column_name
WHERE condition;
```

Joins (cont'd)

- FULL OUTER JOIN
  - Example

```
SELECT Company.Name, Supply.ProductID
FROM Company FULL OUTER JOIN Supply
ON Company.CompanyID = Supply.CompanyID
```

Joins (cont'd)

- Self Join
  - A self join is a regular join, but the table is joined with itself.
  - Syntax

```
SELECT column_name(s)
FROM table1 T1, table1 T2
WHERE condition;
```

  - Example

```
SELECT A.Name, B.Name
FROM Company A, Company B
WHERE A.CompanyID <> B.CompanyID
AND A.Address = B.Address
```

UNION Operator

- The UNION operator is used to combine the result-set of two or more SELECT statements.
  - Every SELECT statement within UNION must have the same number of columns
  - The columns must also have similar data types
  - The columns in every SELECT statement must also be in the same order
- Syntax

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

- The UNION operator selects only distinct values by default. To allow duplicate values, use UNION ALL

UNION Operator (cont'd)

- Example

```
SELECT CompanyID FROM Supply
WHERE ProductID = 1
UNION
SELECT CompanyID FROM Supply
WHERE ProductID = 2
```

INTERSECT Operator

- Return to the result-set that appear in both tables
- Example

```
SELECT CompanyID FROM Supply
WHERE ProductID = 1
INTERSECT
SELECT CompanyID FROM Supply
WHERE ProductID = 2
```

EXCEPT Operator

- Return to the result-set that appear in the first table and not in the second
- Example

```
SELECT CompanyID FROM Supply
WHERE ProductID = 1
EXCEPT
SELECT CompanyID FROM Supply
WHERE ProductID = 2
```

GROUP BY Statement

- The GROUP BY statement groups rows that have the same values into summary rows
- The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.
- Syntax

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

```
SELECT COUNT(CompanyID), Address
FROM Company
GROUP BY Address
```

GROUP BY Statement (cont'd)

- HAVING Clause
  - The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.
  - Syntax

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

  - Example

```
SELECT COUNT(CompanyID), Address
FROM Company
GROUP BY Address
HAVING COUNT(CompanyID)>1
```

EXISTS Operator

- The EXISTS operator is used to test for the existence of any record in a subquery.
- The EXISTS operator returns TRUE if the subquery returns one or more records.
- Syntax

```
SELECT column_name(s)
FROM table_name
WHERE EXISTS
(SELECT column_name FROM table_name WHERE condition);
```

EXISTS Operator (cont'd)

- Example

```
SELECT Company.Name
FROM Company
WHERE EXISTS(SELECT ProductID FROM Supply
WHERE Company.CompanyID = Supply.CompanyID)
```

ANY and ALL Operators

- The ANY and ALL operators allow you to perform a comparison between a single column value and a range of other values.
- The ANY operator:
  - returns a boolean value as a result
  - returns TRUE if ANY of the subquery values meet the condition
  - ANY means that the condition will be true if the operation is true for any of the values in the range.
  - Syntax

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator ANY
  (SELECT column_name
  FROM table_name
  WHERE condition);
```

ANY and ALL Operators (cont'd)

- The ALL operator:
  - returns a boolean value as a result
  - returns TRUE if ALL of the subquery values meet the condition
  - is used with SELECT, WHERE and HAVING statements
  - ALL means that the condition will be true only if the operation is true for all values in the range.
  - Syntax

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator ALL
  (SELECT column_name
  FROM table_name
  WHERE condition);
```

CASE Statement

- The CASE statement goes through conditions and returns a value when the first condition is met (like an if-then-else statement). So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the ELSE clause.
- If there is no ELSE part and no conditions are true, it returns NULL.
- Syntax

```
CASE
    WHEN condition1 THEN result1
    WHEN condition2 THEN result2
    WHEN conditionN THEN resultN
    ELSE result
END;
```

CASE Statement (cont'd)

- Example

```
SELECT CompanyID, ProductID,
CASE
WHEN Quantity <1000 THEN 'Small'
WHEN Quantity <3000 THEN 'Medium'
ELSE 'Large'
END
FROM Supply
```

SQL Operators

- Arithmetic Operators

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

```
SELECT 2 + 3
```

SQL Operators (cont'd)

- Bitwise Operators

Operator	Description
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR

```
SELECT 2 | 3
```

SQL Operators (cont'd)

- Comparison Operators

Operator	Description
=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
<>	Not equal to

SQL Operators (cont'd)

- Logical Operators

Operator	Description
ALL	TRUE if all of the subquery values meet the condition
AND	TRUE if all the conditions separated by AND is TRUE
ANY	TRUE if any of the subquery values meet the condition
BETWEEN	TRUE if the operand is within the range of comparisons
EXISTS	TRUE if the subquery returns one or more records
IN	TRUE if the operand is equal to one of a list of expressions
LIKE	TRUE if the operand matches a pattern
NOT	Displays a record if the condition(s) is NOT TRUE
OR	TRUE if any of the conditions separated by OR is TRUE
SOME	TRUE if any of the subquery values meet the condition

Functions

- MIN() and MAX()
- COUNT(), AVG() and SUM()

Functions (cont'd)

- MIN() and MAX()
  - The MIN() function returns the smallest value of the selected column.
  - The MAX() function returns the largest value of the selected column.
  - Syntax

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

  - Example

```
SELECT MIN(Quantity)
FROM Supply
```

```
SELECT MAX(Price)
FROM Product
```

Functions (cont'd)

- COUNT(), AVG() and SUM()
  - The COUNT() function returns the number of rows that matches a specified criterion.
  - The AVG() function returns the average value of a numeric column.
  - The SUM() function returns the total sum of a numeric column.
  - Syntax

```
SELECT COUNT/AVG/SUM(column_name)
FROM table_name
WHERE condition;
```

Functions (cont'd)

- COUNT(), AVG() and SUM()
  - Example

```
SELECT COUNT(CompanyID)
FROM Company

SELECT COUNT(CompanyID), Address
FROM Company
GROUP BY Address

SELECT AVG(Quantity)
FROM Supply

SELECT SUM(NumberofEmployee)
FROM Company
WHERE Address LIKE '%Japan%'
```

Practice

Hands-on with the BikeStores sample database and Lecturers-Projects-Participation