

# BASIC QUERIES IN SQL

*Instructor:*



# Learning Goals

**By the end of this lecture  
students should be able to:**

## Sql Insert into Statement

```
INSERT INTO agents VALUES ("A001","Jodi","London",12,"075-1248798");
```

agent_code	agent_name	working_area	commission	phone_no
A001	Jodi	London	12	075-1248798

Table : agents



✓ Describe each data manipulation language (DML) statement

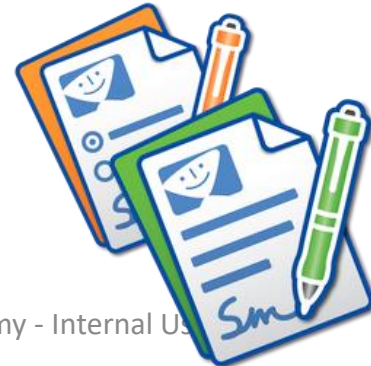
✓ Insert rows into a table

✓ Update rows in a table

✓ Delete rows from a table

# Table of contents

- ✓ INSERT, UPDATE, DELETE Statements
- ✓ SELECT Statement
  - ✓ GROUP BY
  - ✓ HAVING
  - ✓ ORDER BY
- ✓ SQL FUNCTIONS




## Section 1

# INSERT, UPDATE, DELETE

- The **INSERT INTO** statement is used to add one or more rows to a table or a view

## Sql Insert into Statement

```
INSERT INTO agents VALUES ("A001","Jodi","London",.12,"075-1248798");
```



agent_code	agent_name	working_area	commission	phone_no
A001	Jodi	London	.12	075-1248798

Table : agents

# INSERT Statements (2/3)

- **Syntax:**

**(1) Inserting data to all columns**

```
INSERT INTO table_name  
VALUES (value1,value2,value3,...);
```

**Ex1:** `USE Fsoft_Training`  
`INSERT INTO` `dbo.Persons`  
`VALUES ( 1,'Tom', 'B. Erichsen','Skagen 21','Stavanger')`

**(2) Inserting data to selected columns**

```
INSERT INTO table_name(column1,column2,column3,...)  
VALUES (value1,value2,value3,...);
```

**Ex2:** `USE Fsoft_Training`  
`INSERT INTO` `dbo.Customer (CustomerName, City, Country)`  
`VALUES ('Cardinal', 'Stavanger', 'Norway');`

- **Always check data in various cases:**
  - ✓ Normal/Abnormal
  - ✓ Invalid data type
  - ✓ Special characters: ~!@#\$%^&\*()\_
  - ✓ Special string characters: ' char(10) char(13) tab space
  - ✓ Max length, Max/Min value
  - ✓ Duplicated value in UNIQUE constraints

# UPDATE Statement (1/2)

- The **UPDATE** statement is used to changes existing data in a table or view

```
SQLQuery1.sql
1 SELECT TOP 5 * FROM Sales.CurrencyRate
2 GO
3
4 UPDATE Sales.CurrencyRate
5 SET AverageRate = AverageRate + 0.01,
6     EndOfDayRate = EndOfDayRate + 0.01
7 GO
8
9 SELECT TOP 5 * FROM Sales.CurrencyRate
10
```

	CurrencyRateID	CurrencyRateDate	FromCurrencyCode	ToCurrencyCode	AverageRate	EndOfDayRate
1	1	2001-07-01 00:00:00.000	USD	ARS	1.00	1.0002
2	2	2001-07-01 00:00:00.000	USD	AUD	1.5491	1.55
3	3	2001-07-01 00:00:00.000	USD	BRL	1.9379	1.9419
4	4	2001-07-01 00:00:00.000	USD	CAD	1.4641	1.4683
5	5	2001-07-01 00:00:00.000	USD	CNY	8.2781	8.2784

	CurrencyRateID	CurrencyRateDate	FromCurrencyCode	ToCurrencyCode	AverageRate	EndOfDayRate
1	1	2001-07-01 00:00:00.000	USD	ARS	1.01	1.0102
2	2	2001-07-01 00:00:00.000	USD	AUD	1.5591	1.56
3	3	2001-07-01 00:00:00.000	USD	BRL	1.9479	1.9519
4	4	2001-07-01 00:00:00.000	USD	CAD	1.4741	1.4783
5	5	2001-07-01 00:00:00.000	USD	CNY	8.2881	8.2884

- Best Practice**

- ✓ Use the **@@ROWCOUNT** function to return the number of inserted rows to the client application.



# UPDATE Statement (2/2)

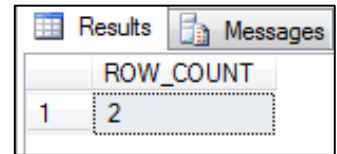
- **Syntax:**

```
UPDATE table_name  
SET column1=value1,column2=value2,...  
WHERE some_column=some_value;
```

**Notice the WHERE clause in the SQL UPDATE statement!**

The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

- **Ex:**      `USE Fsoft_Training`  
             `UPDATE dbo.Customer`  
             `SET PostalCode = '4006'`  
             `WHERE Country = 'Norway'`  
             `SELECT @@ROWCOUNT AS ROW_COUNT`



Results	
ROW_COUNT	
1	2

# DELETE Statement (1/2)

- Removes one or more rows from a table or view

CustomerId	CustomerName	ContactName
1	Alfreds Futterkiste	Maria Anders
2	Around the Horn	Thomas Hardy
3	Berglunds snabbköp	Christina Berglund
4	Antonio Moreno	Antonio Moreno
5	Ana Trujillo	Ana Trujillo

- Best Practice:**

To delete all the rows in a table, use TRUNCATE TABLE. TRUNCATE TABLE is faster than DELETE and uses fewer system and transaction log resources.

TRUNCATE TABLE has restrictions, for example, the table cannot participate in replication



# DELETE Statement (2/2)

- **Syntax:**

```
DELETE FROM table_name  
WHERE some_column=some_value;
```

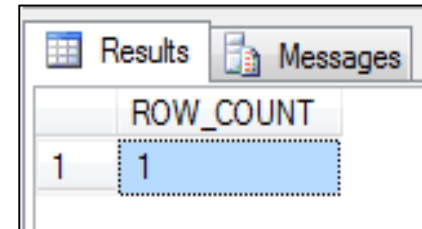
- **Notice the WHERE clause in the SQL DELETE statement!**

The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

Please note that the DELETE FROM command cannot delete any rows of data that would violate FOREIGN KEY or other constraints.

- **Ex:**

```
USE Fsoft_Training  
DELETE dbo.Customer  
WHERE Country = 'Germany'  
SELECT @@ROWCOUNT AS ROW_COUNT
```



The screenshot shows a SQL Server Enterprise Manager window with two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with one row and one column. The column header is 'ROW\_COUNT' and the value in the row is '1'.

ROW_COUNT
1

## Section 2

# SELECT STATEMENT

# SELECT Statement (1/4)

- Retrieves rows from the database and enables the selection of one or many rows or columns from one or many tables



# SELECT Statement (2/4)

## ■ Syntax:

**SELECT** [ALL/DISTINCT/TOP [ WITH TIES ] ] <Column name1>, <Column name2>

**FROM** <Table name>

[**WHERE** <Search condition>]

[**GROUP BY** grouping columns]

[**HAVING** search condition]

[**ORDER BY** sort specification]

**Ex1:** **USE** AdventureWorks

**GO**

**SELECT** ProductID, Name

**FROM** Production.Product

**ORDER BY** Name **ASC**;

(504 rows)

**Ex2:** **SELECT DISTINCT** E.Title

**FROM** HumanResources.Employee E

**ORDER BY** E.Title;

(67 rows)

ProductID	Name
1	Adjustable Race
2	All-Purpose Bearing
3	AWC Ball Bearing
4	BB Ball Bearing
5	Bearing
6	Bike Wheel
7	Blade
8	Cable
9	Chain
10	Chainring
11	Chainring
12	Chainring
13	Chainring
14	Classic
15	Classic
16	Classic
17	Cone-Sprocket

Title
Accountant
Accounts Manager
Accounts Payable Specialist
Accounts Receivable Specialist
Application Specialist
Assistant to the Chief Financial Officer
Benefits Specialist
Buyer
Chief Executive Officer
Chief Financial Officer
Control Specialist
Database Administrator
Design Engineer
Document Control Assistant
Document Control Manager
Engineering Manager
European Sales Manager

- **SQL Alias syntax:**

- ✓ *For table*

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

- ✓ *For Column(s)*

- ```
SELECT column_name AS alias_name
FROM table_name
```

- **Ex:**

- ```
USE AdventureWorks
GO
SELECT c.CustomerID, s.Name
FROM Sales.Customer AS c
JOIN Sales.Store AS s
ON c.CustomerID = s.SalesPersonID
```

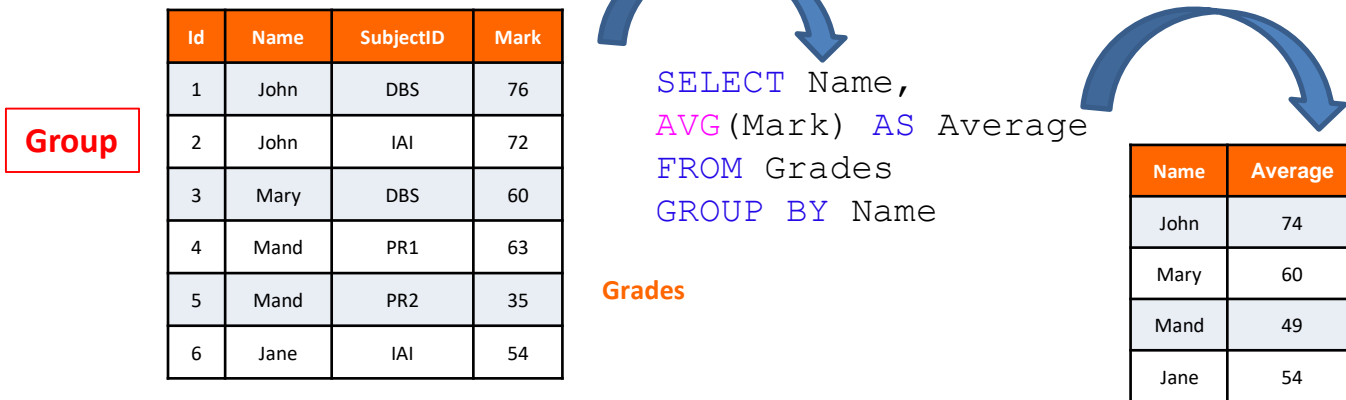
# Grouping by clause

- Sometimes we want to apply aggregate functions to groups of rows.

## Syntax:

```
SELECT column_name, aggregate_function(column_name)
FROM table_name
WHERE column_name operator value
GROUP BY column_name;
```

- Example, find the average mark of each student.





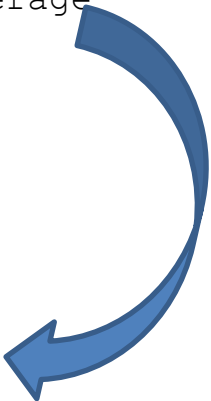
# Having clause

- **HAVING** is like a **WHERE** clause, except that it applies to the results of a **GROUP BY** query.
- It can be used to select groups which satisfy a given condition.
- **Ex:**



| Id | Name | SubjectID | Mark |
|----|------|-----------|------|
| 1  | John | DBS       | 76   |
| 2  | John | IAI       | 72   |
| 3  | Mary | DBS       | 60   |
| 4  | Mand | PR1       | 63   |
| 5  | Mand | PR2       | 35   |
| 6  | Jane | IAI       | 54   |

```
SELECT Name, AVG(Mark) AS Average
FROM Grades
GROUP BY Name
HAVING AVG(Mark) >= 50
```




| Name | Average |
|------|---------|
| John | 74      |
| Mary | 60      |
| Jane | 54      |


# WHERE and HAVING

- **WHERE** refers to the rows of tables, and so cannot use aggregate functions
- **HAVING** refers to the groups of rows, can use aggregate functions and cannot use columns which are not in the GROUP BY

```
SELECT Name,  
AVG(Mark) AS Average  
FROM Grades  
WHERE AVG(Mark) >= 50  
GROUP BY Name
```



```
SELECT Name,  
AVG(Mark) AS Average  
FROM Grades  
GROUP BY Name  
HAVING AVG(Mark) >= 50
```



# Order by clause

- The SQL **ORDER BY clause** is used to sort (ascending or descending) the records in the result set for a SELECT statement.

## Syntax:

```
SELECT column_name, column_name  
FROM table_name  
[WHERE conditions]  
ORDER BY column_name, column_name [ASC | DESC]
```

## Ex:

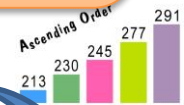
Group

| Id | Name | SubjectID | Mark |
|----|------|-----------|------|
| 1  | John | DBS       | 76   |
| 2  | John | IAI       | 72   |
| 3  | Mary | DBS       | 60   |
| 4  | Mand | PR1       | 63   |
| 5  | Mand | PR2       | 35   |
| 6  | Jane | IAI       | 54   |

```
SELECT Name,  
AVG(Mark) AS Average  
FROM Grades  
GROUP BY Name  
ORDER BY Average DESC
```

Grades

| Name | Average |
|------|---------|
| John | 74      |
| Mary | 60      |
| Jane | 54      |
| Mand | 49      |



## Section 3

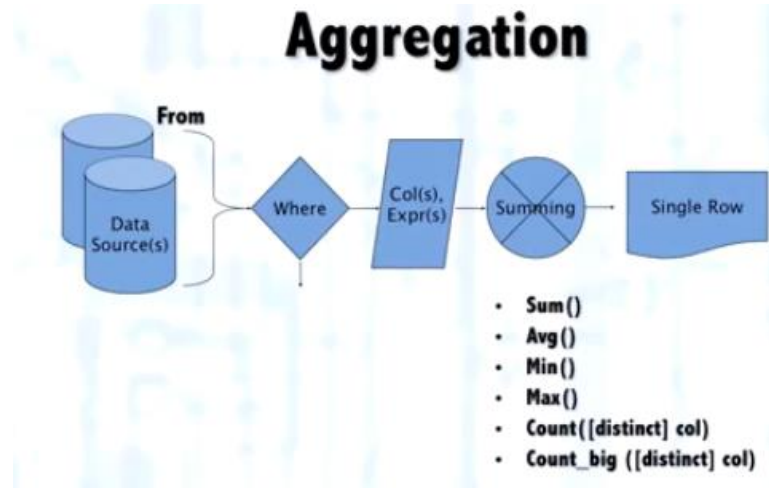
# SQL FUNCTIONS

- SQL has many built-in functions for performing calculations on data:
  - ✓ SQL aggregate functions return a single value, calculated from values in a column.
  - ✓ SQL scalar functions return a single value, based on the input value.



# What is an aggregate function

- An **aggregate function** is function that take a collection of values as input and return a single value.
- Aggregate functions can be used as expressions only in the following:
  - ✓ The select list of a SELECT statement
  - ✓ A HAVING clause.



# Aggregate Functions

- Each function eliminates NULL values and operates on Non-NULL values

| Function | Description                                         |
|----------|-----------------------------------------------------|
| AVG ()   | Return the average value in a column                |
| COUNT()  | Return the total number of values in a given column |
| COUNT(*) | Return the number of rows                           |
| MIN ()   | Returns the smallest value in a column              |
| MAX ()   | Returns the largest value in a column               |
| SUM()    | Returns the sum values in a column                  |

# Scalar functions

| Function | Description                                                |
|----------|------------------------------------------------------------|
| LEN()    | Returns the length of a text field                         |
| ROUND()  | Rounds a numeric field to the number of decimals specified |
| NOW()    | Returns the current system date and time                   |
| FORMAT() | Formats how a field is to be displayed                     |



# UNION Operator

- The SQL UNION operator combines the result of two or more SELECT statements.

## Syntax:

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



**Note:** The UNION operator selects only distinct values by default. To allow duplicate values, use the **ALL** keyword with UNION.

```
SELECT Column1, Column2 FROM Table1  
UNION  
SELECT Column1, Column2 FROM  
Table2;
```

| Table 1  |          |
|----------|----------|
| Column 1 | Column 2 |
| a        | a        |
| a        | b        |
| a        | c        |

UNION

| Table 2  |          |
|----------|----------|
| Column 1 | Column 2 |
| b        | a        |
| a        | b        |
| b        | c        |

Result

| Column 1 | Column 2 |
|----------|----------|
| a        | a        |
| a        | b        |
| a        | c        |
| b        | a        |
| b        | c        |

The UNION operator selects only distinct values by default.

Duplicate rows are displayed only once.

```
SELECT Column1, Column2 FROM Table1  
UNION ALL  
SELECT Column1, Column2 FROM  
Table2;
```

| Table 1  |          |
|----------|----------|
| Column 1 | Column 2 |
| a        | a        |
| a        | b        |
| a        | c        |

UNION  
ALL

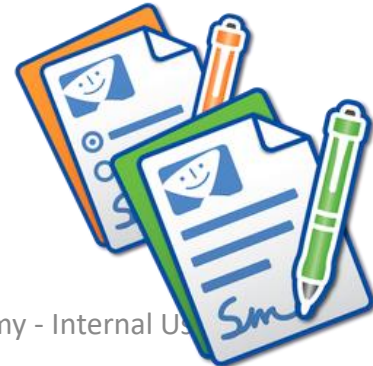
| Table 2  |          |
|----------|----------|
| Column 1 | Column 2 |
| b        | a        |
| a        | b        |
| b        | c        |

Result

| Column 1 | Column 2 |
|----------|----------|
| a        | a        |
| a        | b        |
| a        | b        |
| a        | c        |
| b        | a        |
| b        | c        |

Duplicate rows are repeated in the result set.

- ✓ INSERT, UPDATE, DELETE Statements
- ✓ SELECT Statement
  - ✓ GROUP BY
  - ✓ HAVING
  - ✓ ORDER BY
- ✓ SQL FUNCTIONS



# Thank you

