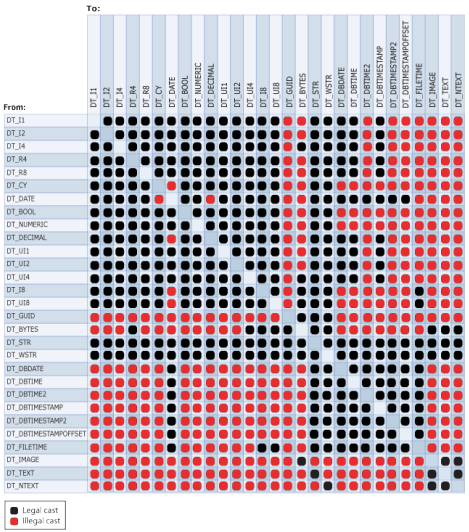
# BUILT-IN FUNCTIONS

*Instructor:*

# Learning Goals

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

- Understand about the built-in functions in SQL Server
- Recognize how to use built-in functions to perform operations on data




```
SELECT SUBSTRING('sunil',3,5)

SELECT SUBSTRING('SQLServer2008',10,4)

DECLARE @Name VARCHAR(20)
SET @Name='Sunil Kumar'
SELECT
SUBSTRING(@Name,1,CHARINDEX(' ',@Name)-1)AS FirstName,
SUBSTRING(@Name,CHARINDEX(' ',@Name)+1,LEN(@Name))AS LastName
```

Results | Messages

| | (No column name) |
|---|---|
| 1 | nil |

| | (No column name) |
|---|---|
| 1 | 2008 |

| | FirstName | LastName |
|---|---|---|
| 1 | Sunil | Kumar |

# Table of contents

- Conversion Functions
- Date and Time Functions
- String Functions

Section1

# CONVERSION FUNCTIONS

# CAST Function

- Converts an expression of one data type to another in SQL Server 2008 R2.

**Syntax for CAST:**

```
CAST ( expression AS data_type [ ( length ) ] )
```

- The **Cast**() function is used to convert a data type variable or data from one data type to another data type.
- The **Cast**() function provides a data type to a dynamic parameter (?) or a NULL value.

# CONVERT Function (1/3)

- When you convert expressions from one type to another, in many cases there will be a need within a stored procedure or other routine to convert data from a **datetime type** to a **varchar** type.
- The Convert function is used for such things. The CONVERT() function can be used to **display date/time data in various formats**

**Syntax for CONVERT:**

```
CONVERT ( data_type [ ( length ) ] , expression [ , style ] )
```

✓ Style (0 hoặc 100): **mon dd yyyy hh:miAM (or PM)**

# CONVERT Function (2/3)

| Without century (yy) | With century (yyyy) | Standard | Input/Output |
|---|---|---|---|
| - | 0 or 100 | Default | mon dd yyyy hh:miAM (or PM) |
| 1 | 101 | U.S. | mm/dd/yyyy |
| 2 | 102 | ANSI | yy.mm.dd |
| 3 | 103 | British/French | dd/mm/yyyy |
| 4 | 104 | German | dd.mm.yy |
| 5 | 105 | Italian | dd-mm-yy |
| 6 | 106 | - | dd mon yy |
| 7 | 107 | - | Mon dd, yy |
| 8 | 108 | - | hh:mi:ss |
| - | 9 or 109 | Default + milliseconds | mon dd yyyy hh:mi:ss:mmmAM (or PM) |
| 10 | 110 | USA | mm-dd-yy |
| 11 | 111 | JAPAN | yy/mm/dd |

# CONVERT Function (3/3)

| Without century (yy) | With century (yyyy) | Standard | Input/Output |
|---|---|---|---|
| 12 | 112 | ISO | yymmdd<br>Yyyymmdd |
| - | 13 or 113 | Europe default + milliseconds | dd mon yyyy hh:mi:ss:mmm(24h) |
| 14 | 114 | - | hh:mi:ss:mmm(24h) |
| - | 20 or 120 | ODBC canonical | yyyy-mm-dd hh:mi:ss(24h) |
| - | 21 or 121 | ODBC canonical (with milliseconds) | yyyy-mm-dd hh:mi:ss.mmm(24h) |
| - | 126 | ISO8601 | yyyy-mm-ddThh:mi:ss.mmm (no spaces) |
| - | 127 | ISO8601 with time zone Z | yyyy-mm-ddThh:mi:ss.mmmZ<br>(no spaces) |
| - | 130 | Hijri | dd mon yyyy hh:mi:ss:mmmAM |
| - | 131 | Hijri | dd/mm/yy hh:mi:ss:mmmAM |

Section2

# DATE AND TIME FUNCTIONS

# GETDATE() & DATEPART() Function (1/2)

- The **GETDATE**() function returns the current date and time from the SQL Server.
- The **DATEPART**() function is used to return a single part of a date/time, such as year, month, day, hour, minute, etc.

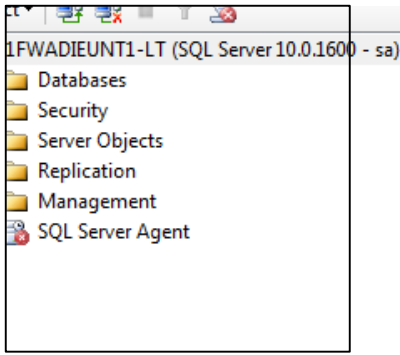
**Syntax:**

```
GETDATE()
DATEPART(datepart, date)
```

- **Ex** :

```
SELECT GETDATE()
SELECT DATEPART(YYYY, GETDATE())
```

**Result** :



# GETDATE() & DATEPART Function (2/2)

| datepart | Abbreviation |
|---|---|
| year | yy, yyyy |
| quarter | qq, q |
| month | mm, m |
| dayofyear | dy, y |
| day | dd, d |
| week | wk, ww |
| weekday | dw, w |
| hour | hh |
| minute | mi, n |
| second | ss, s |
| millisecond | ms |
| microsecond | mcs |
| nanosecond | ns |

# DAY, MONTH,YEAR Function

- Returns an integer representing the day/month/year (day of the month) of the specified *date*.

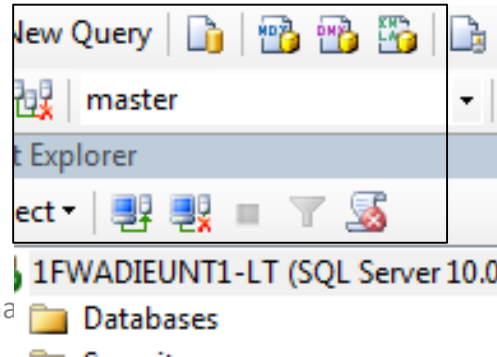
**Syntax:**

```
DAY(date)
MONTH(date)
YEAR(date)
```

- **Ex** : 

```
SELECT DAY(GETDATE()) AS [Day],
MONTH(GETDATE()) AS [Month],
YEAR(GETDATE()) AS [Year]
```

**Result** :



# DATEADD Function

- The **DATEADD**() function adds or subtracts a specified time interval from a date.

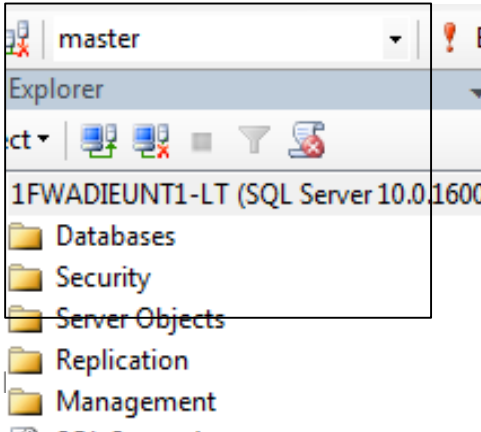
**Syntax:**

```
DATEADD(datepart,number,date)
```

- **Ex** :

```
DECLARE @dt datetime
SET @dt = GETDATE()
SELECT @dt AS CurrentDate
SELECT DATEADD(day, 30, @dt) AS AffterDate
```

**Result** :



# DATEDIFF Function

- The **DATEDIFF**() function returns the time between two dates.

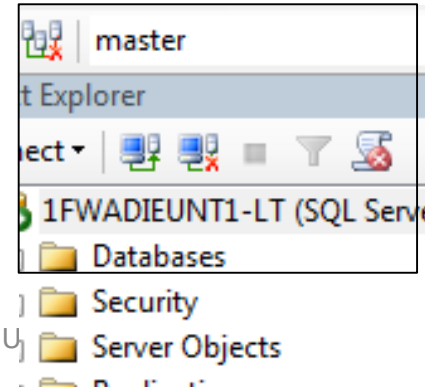
**Syntax:**

```
DATEDIFF (datepart,startdate,enddate)
```

- **Ex:**

```
DECLARE @date1 DATETIME
DECLARE @date2 DATETIME
SET @date1= '2012-04-07 20:12:22.013'
SET @date2= '2014-02-27 22:14:10.013'
SELECT DATEDIFF(month, @date1, @date2) AS 'Month'
```

**Result**:



Section3

# STRING FUNCTIONS

# RTRIM, LTRIM Function

- LTRIM Removes all white spaces from the beginning of the string.

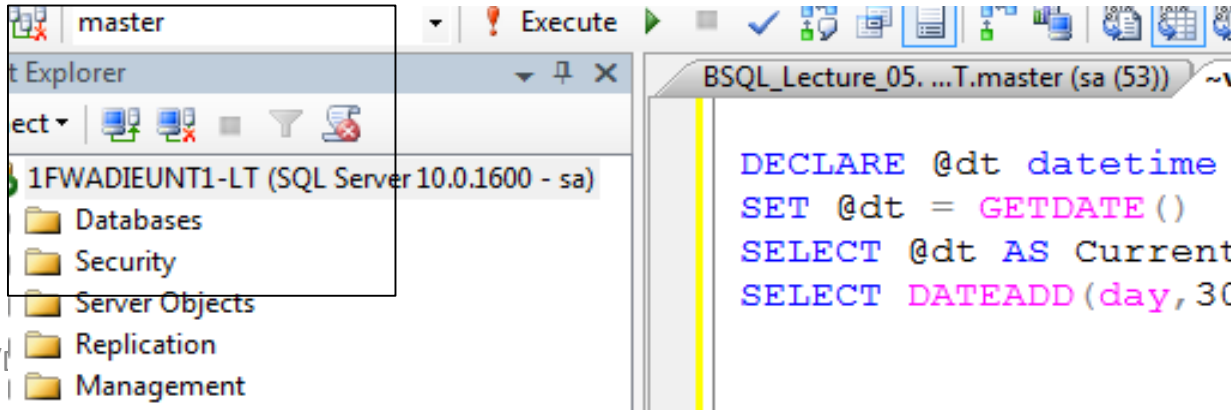
**Syntax:**

```
LTRIM (str)
RTRIM (str)
```

- **Ex** : 

```
SELECT LTRIM('   Sample   ');
SELECT RTRIM('   Sample   ');
```

**Result** :



# SUBSTRING Function

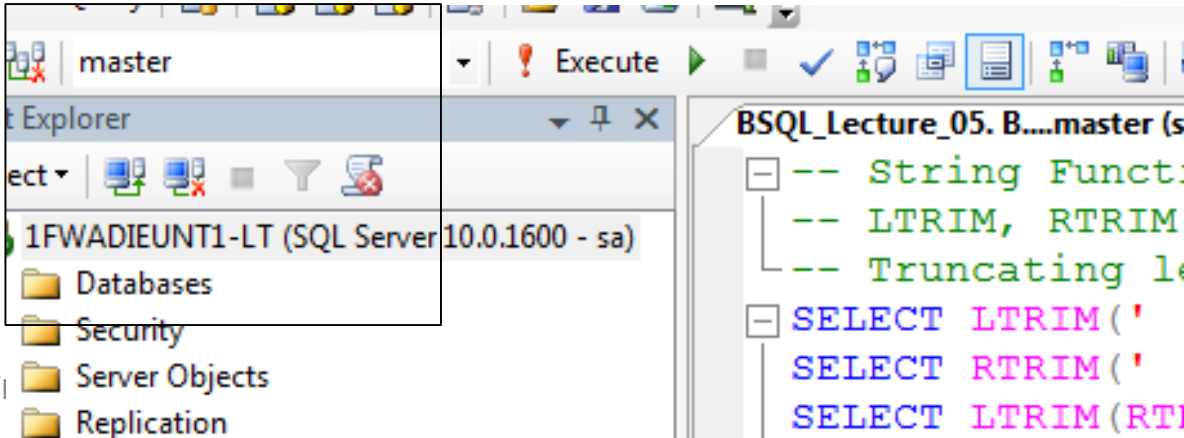
- The **Substring** function in SQL is used to return a portion of string. This function is called differently in different databases:

**Syntax:**

```
SUBSTRING(str, position, length)
```

- **Ex** : SELECT SUBSTRING('Bill Gates', 0 ,5) As Result

**Result** :



# LEN, CHARINDEX, PATINDEX Function

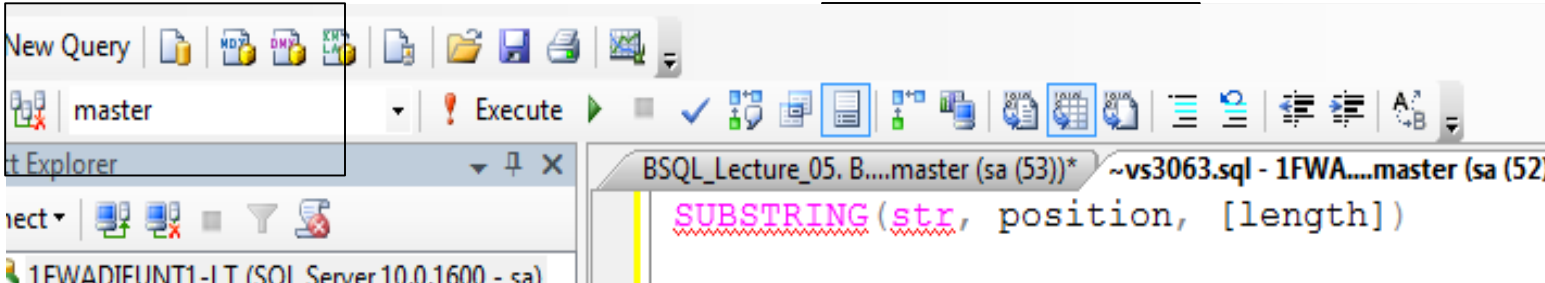
- The **CHARINDEX** and **PATINDEX** functions return the starting position of a pattern you specify.
- PATINDEX can use **wildcard characters**, but CHARINDEX cannot

**Syntax:** LEN(str)
CHARINDEX ( expression1 ,expression2 [ , start_location ] )
PATINDEX ( '%pattern%' , expression )

- **Ex :**

```
SELECT CHARINDEX('bicycle',
        'Reflectors are vital safety components of your bicycle.') AS Positions
SELECT PATINDEX ( '%ein%', 'Das ist ein Test') AS Positions
```

**Result**:



# Summary

- ✓ Conversion Functions
  - CAST, CONVERT Function
- ✓ Date and Time Functions
  - GETDATE, DATEPART, DAY, MONTH,YEAR, DATEDD, DATEIFF Function
- ✓ String Functions
  - RTRIM, LTRIM, SUBSTRING, LEN, CHARINDEX, PATINDEX
- ✓ Demo

# Thank you