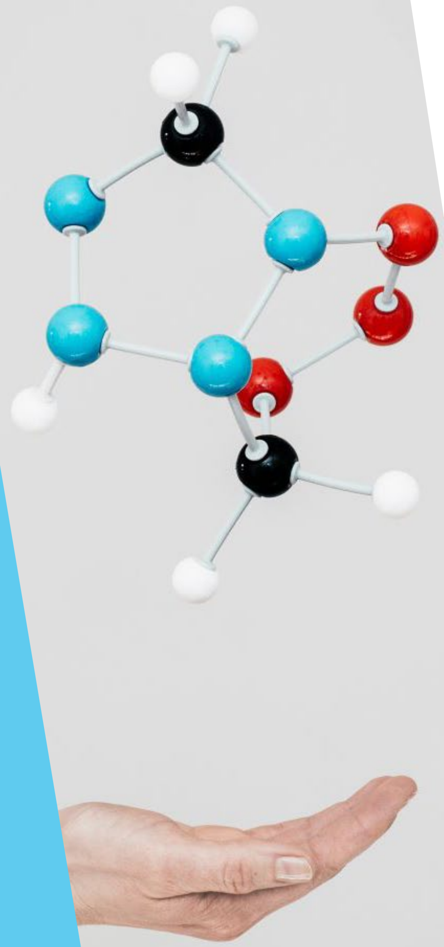
CMPE 2400 Databases

Built In Functions

Built In Functions

- In C#, and the rest of the .NET languages, you have access to the Base Class Library (BCL).
- SQL does not use the BCL, but there is some built-in system support through globally available functions.
- The return values from these functions may be used in place of variables or literal values in calculations and even as embedded function calls.



Function Classification

- Deterministic
  - Returns the same result each time the function is called given:
    - The input values are the same
    - The database is in the same state
- Non-Deterministic
  - May return a different result even when the input values and the database are in the same state

Function Classification

- In addition to being deterministic or non-deterministic, a function is classified in one of the following categories:
  - Scalar functions
    - Operate on a single value and return a single value
    - May be used wherever an expression is valid
  - Aggregate functions
    - Operate on a collection of values, but return a single, summarizing value
  - Rowset functions (not covered here)
    - Return an object that can be used like table references in an SQL statement
  - Ranking functions (not covered here)
    - Return a ranking value for each row in a partition

Scalar Functions

- Mathematical Functions
  - Perform calculations based on input values provided as parameters to the functions, and return numeric values
- String Functions
  - Perform operations on a string (char or varchar) input value and return a string or numeric value
- Date and Time Functions
  - Perform operations on a date and time input value and return string, numeric, or date and time values
- Miscellaneous Functions
  - This group will cover things such as casting and value substitution

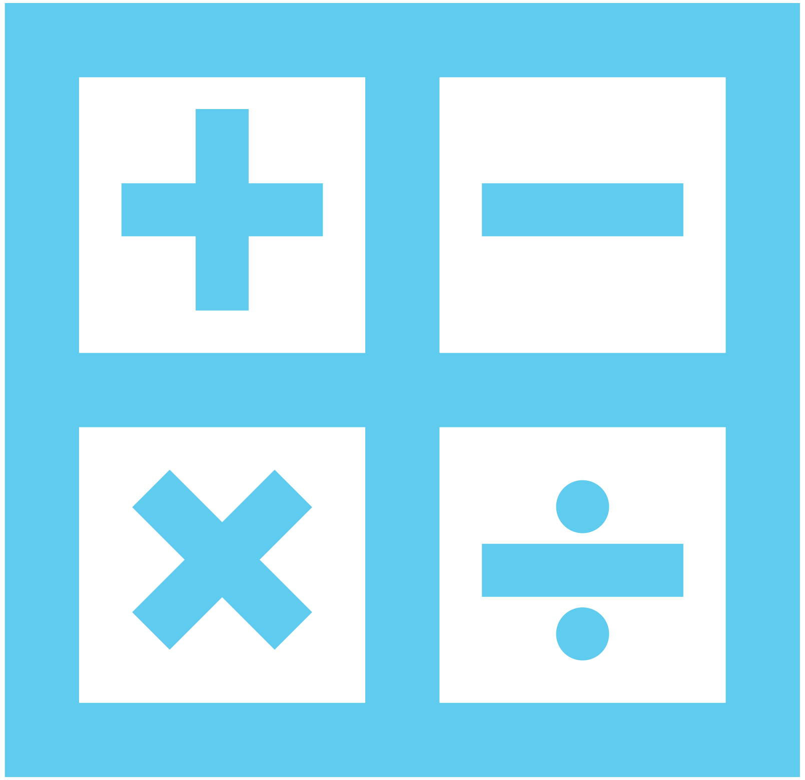
Scalar Functions

- Not covered in detail in this course:
  - Configuration Functions
    - Return information about the current system configuration
  - Cursor Functions
    - Return information about cursors
  - Metadata Functions
    - Return information about the database and database objects
  - Security Functions
    - Return information about users and roles
  - System Functions
    - Perform operations and return information about values, objects, and settings in an instance of SQL Server
  - System Statistical Functions
    - Return statistical information about the system
  - Text and Image Functions
    - Perform operations on text or image input values or columns, and return information about the value

Mathematical Functions

- Mathematical functions may generally be divided into two groups
  - Algebraic Functions
  - Trigonometric Functions

- Aside from the rand() function, all built-in mathematical functions are deterministic

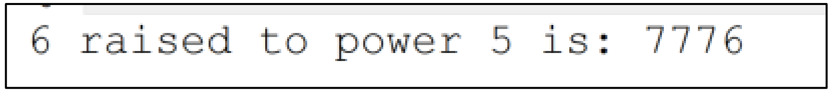


Math Functions

- ABS
- ACOS
- ASIN
- ATAN
- ATN2
- CEILING
- COS
- COT
- DEGREES
- EXP
- FLOOR
- LOG
- LOG10
- PI
- POWER
- RADIANS
- RAND
- ROUND
- SIGN
- SIN
- SQRT
- SQUARE
- TAN

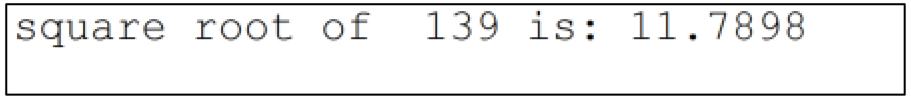
Exercise

- Output the following using SSMS:
  - 6 raised to the power 5



```
6 raised to power 5 is: 7776
```

  - Square root of 139 to 4 decimal places



```
square root of  139 is: 11.7898
```

String Functions

Scalar functions that perform an operation on an input string value of type char, varchar, nchar, nvarchar, binary, or varbinary

All built-in string functions are deterministic

It is important to remember that SQL string types are considered to be **1 Indexed** as opposed to the 0 Indexed string types seen in many other programming languages

Return either a numerical or string value

String Functions

ASCII	CHAR	CHARINDEX	CONCAT	CONCAT_WS
DIFFERENCE	FORMAT	LEFT	LEN	LOWER
LTRIM	NCHAR	PATINDEX	QUOTENAME	REPLACE
REPLICATE	REVERSE	RIGHT	RTRIM	SOUNDEX
SPACE	STR	STRING_AGG	STRING_ESCAPE	STRING_SPLIT
STUFF	SUBSTRING	TRANSLATE	TRIM	UNICODE

Exercise 2

- Write suitable SQL statements to display the following:

1. The ASCII code of the character ‘T’
2. The character with ASCII code 100
3. The character with Unicode 25000
4. The number of characters in the string ‘This is a string’
5. The number of bytes used to store the string ‘This is a string’ as nvarchar
6. The starting position of the string ‘story’ in the string ‘This is a long story’

Exercise 3

- Write suitable SQL statements to display the following:

1. The substring of length 5 starting at position 8 in ‘We need to get this work done’.
2. The leftmost 12 characters of the string ‘This is a rather long string’.
3. The rightmost 10 characters in the string ‘This is a rather long string’.
4. The string obtained when all occurrences of ‘Stevenson’ is replaced by ‘farmers’ in the sentence ‘The Company Stevenson, Stevenson, Stevenson and Stevenson consists of 4 brothers from the Stevenson family’

Date and Time Functions

- Scalar functions that perform an operation on a date and time input value and return a string, numeric, or date and time value

- Not all built-in datetime functions are deterministic in nature

Date and Time Functions

- *dateparts* which make up the **datetime** data types:

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

- When smalldatetime values are used, seconds and milliseconds are always considered to be 0
- Using 4 digit years is recommended

Date and Time Datatypes

Data type	Format	Range	Accuracy	Storage size (bytes)	User-defined fractional second precision	Time zone offset
time	hh:mm:ss[.nnnnnnn]	00:00:00.0000000 through 23:59:59.9999999	100 nanoseconds	3 to 5	Yes	No
date	YYYY-MM-DD	0001-01-01 through 9999-12-31	1 day	3	No	No
smalldatetime	YYYY-MM-DD hh:mm:ss	1900-01-01 through 2079-06-06	1 minute	4	No	No
datetime	YYYY-MM-DD hh:mm:ss[.nnn]	1753-01-01 through 9999-12-31	0.00333 second	8	No	No
datetime2	YYYY-MM-DD hh:mm:ss[.nnnnnnn]	0001-01-01 00:00:00.0000000 through 9999-12-31 23:59:59.9999999	100 nanoseconds	6 to 8	Yes	No
datetimeoffset	YYYY-MM-DD hh:mm:ss[.nnnnnnn] [+|-]hh:mm	0001-01-01 00:00:00.0000000 through 9999-12-31 23:59:59.9999999 (in UTC)	100 nanoseconds	8 to 10	Yes	Yes

System Date and Time Functions

Function	Syntax	Return value	Return data type	Determinism	Precision
SYSDATETIME	SYSDATETIME ( )	Returns a **datetime2(7)** value containing the date and time of the computer on which the instance of SQL Server runs. The returned value does not include the time zone offset.	**datetime2(7)**	Nondeterministic	Higher
SYSDATETIMEOFFSET	SYSDATETIMEOFFSET ( )	Returns a **datetimeoffset(7)** value containing the date and time of the computer on which the instance of SQL Server runs. The returned value includes the time zone offset.	**datetimeoffset(7)**	Nondeterministic	Higher
SYSUTCDATETIME	SYSUTCDATETIME ( )	Returns a **datetime2(7)** value containing the date and time of the computer on which the instance of SQL Server is running. The function returns the date and time values as UTC time (Coordinated Universal Time).	**datetime2(7)**	Nondeterministic	Higher
CURRENT_TIMESTAMP	CURRENT_TIMESTAMP	Returns a **datetime** value containing the date and time of the computer on which the instance of SQL Server runs. The returned value does not include the time zone offset.	**datetime**	Nondeterministic	Lower
GETDATE	GETDATE ( )	Returns a **datetime** value containing the date and time of the computer on which the instance of SQL Server runs. The returned value does not include the time zone offset.	**datetime**	Nondeterministic	Lower
GETUTCDATE	GETUTCDATE ( )	Returns a **datetime** value containing the date and time of the computer on which the instance of SQL Server runs. The function returns the date and time values as UTC time (Coordinated Universal Time).	**datetime**	Nondeterministic	Lower

Returning Date and Time Parts

Function	Syntax	Return value	Return data type	Determinism
DATENAME	DATENAME (*datepart* , *date*)	Returns a character string representing the specified *datepart* of the specified date.	**nvarchar**	Nondeterministic
DATEPART	DATEPART (*datepart* , *date*)	Returns an integer representing the specified *datepart* of the specified *date*.	**int**	Nondeterministic
DAY	DAY (*date*)	Returns an integer representing the day part of the specified *date*.	**int**	Deterministic
MONTH	MONTH (*date*)	Returns an integer representing the month part of a specified *date*.	**int**	Deterministic
YEAR	YEAR (*date*)	Returns an integer representing the year part of a specified *date*.	**int**	Deterministic

Function	Syntax	Return value	Return data type	Determinism
DATEFROMPARTS	DATEFROMPARTS (*year*, *month*, *day*)	Returns a **date** value for the specified year, month, and day.	**date**	Deterministic
DATETIME2FROMPARTS	DATETIME2FROMPARTS (*year*, *month*, *day*, *hour*, *minute*, *seconds*, *fractions*, *precision*)	Returns a **datetime2** value for the specified date and time, with the specified precision.	**datetime2(** *precision* **)**	Deterministic
DATETIMEFROMPARTS	DATETIMEFROMPARTS (*year*, *month*, *day*, *hour*, *minute*, *seconds*, *milliseconds*)	Returns a **datetime** value for the specified date and time.	**datetime**	Deterministic
DATETIMEOFFSETFROMPARTS	DATETIMEOFFSETFROMPARTS (*year*, *month*, *day*, *hour*, *minute*, *seconds*, *fractions*, *hour_offset*, *minute_offset*, *precision*)	Returns a **datetimeoffset** value for the specified date and time, with the specified offsets and precision.	**datetimeoffset(** *precision* **)**	Deterministic
SMALLDATETIMEFROMPARTS	SMALLDATETIMEFROMPARTS (*year*, *month*, *day*, *hour*, *minute*)	Returns a **smalldatetime** value for the specified date and time.	**smalldatetime**	Deterministic
TIMEFROMPARTS	TIMEFROMPARTS (*hour*, *minute*, *seconds*, *fractions*, *precision*)	Returns a **time** value for the specified time, with the specified precision.	**time(** *precision* **)**	Deterministic

Returning Date and Time Values

Function	Syntax	Return value	Return data type	Determinism
DATEDIFF	DATEDIFF (*datepart* , *startdate* , *enddate*)	Returns the number of date or time *datepart* boundaries, crossed between two specified dates.	int	Deterministic
DATEDIFF_BIG	DATEDIFF_BIG (*datepart* , *startdate* , *enddate*)	Returns the number of date or time *datepart* boundaries, crossed between two specified dates.	bigint	Deterministic
DATEADD	DATEADD (datepart , number , date)	Returns a new datetime value by adding an interval to the specified datepart of the specified date.	The data type of the date argument	Deterministic
EOMONTH	EOMONTH (start_date [, month_to_add])	Returns the last day of the month containing the specified date, with an optional offset.	Return type is the type of the start_date argument, or alternately, the date data type.	Deterministic

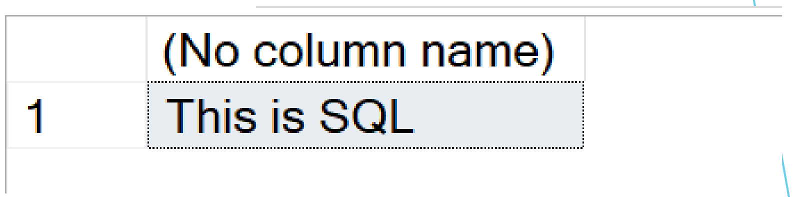
Date and Time Difference and Modification

Using select statements

- Select Statement is generally used to retrieve data from databases and display the data.
- However, they can also be used to display:
    - literal values
    - Values of variables
    - Values of global constants
- With select statements, we can specify column titles for the different values

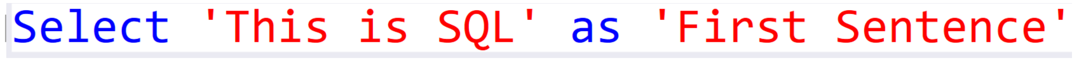
Using Select to display string literals

```sql
Select 'This is SQL'
```

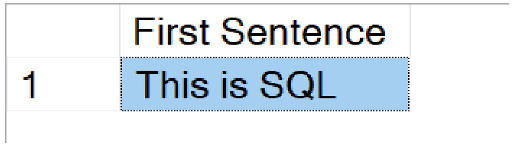


```
     (No column name)
1    This is SQL
```

Displaying a value with a column title

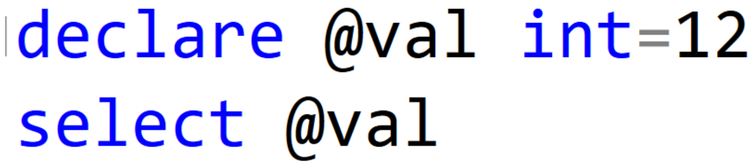


```sql
Select 'This is SQL' as 'First Sentence'
```

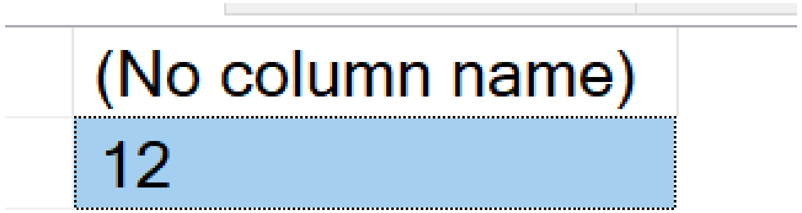


	First Sentence
1	This is SQL

Using Select to display a variable



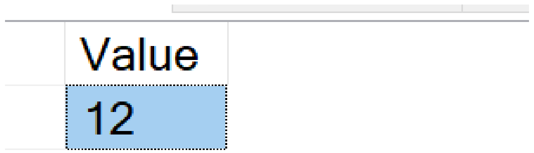
```sql
declare @val int=12
select @val
```



```
(No column name)
12
```

Displaying variable with Title

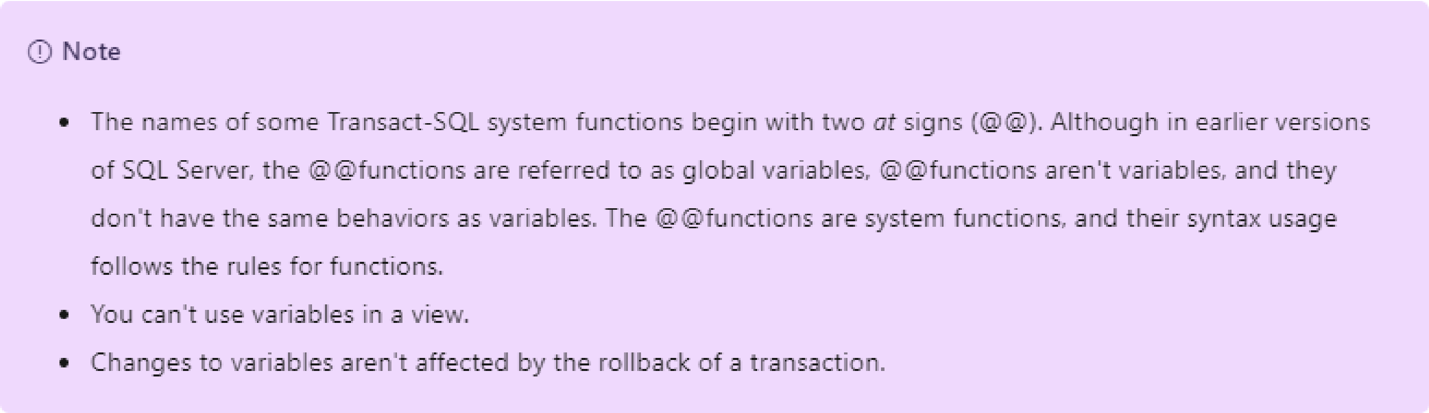
```
declare @val int=12
select @val as Value
```



```
Value
12
```

System Functions

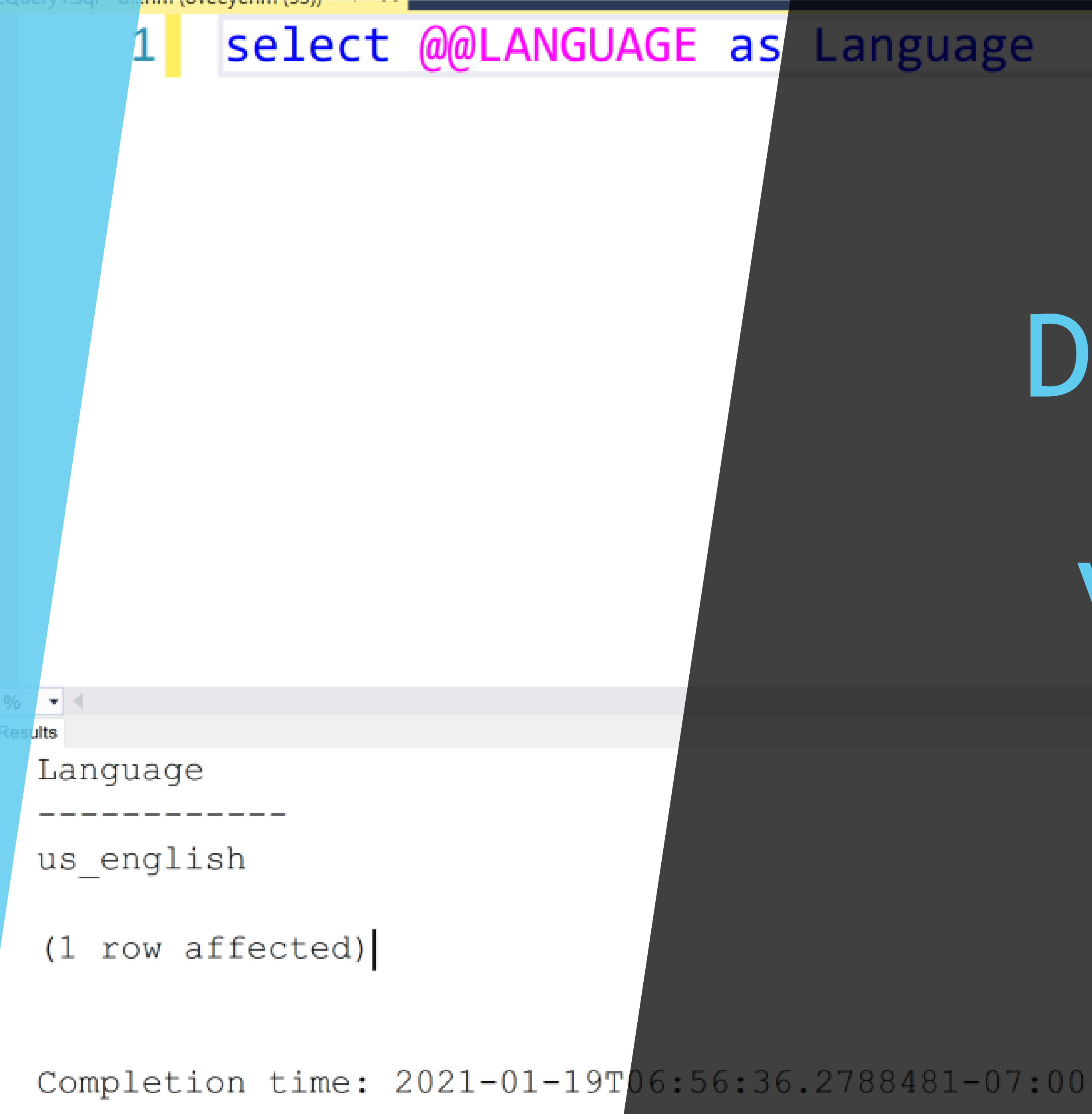
- $PARTITION
- ERROR_PROCEDURE
- @@ERROR
- ERROR_SEVERITY
- @@IDENTITY
- ERROR_STATE
- @@PACK_RECEIVED
- FORMATMESSAGE
- @@ROWCOUNT
- GET_FILESTREAM_TRANSACTION_CONTEXT
- @@TRANCOUNT
- GETANSINULL
- BINARY_CHECKSUM
- HOST_ID
- CHECKSUM
- HOST_NAME
- COMPRESS
- ISNULL
- CONNECTIONPROPERTY
- ISNUMERIC
- CONTEXT_INFO
- MIN_ACTIVE_ROWVERSION
- CURRENT_REQUEST_ID
- NEWID
- CURRENT_TRANSACTION_ID
- NEWSEQUENTIALID
- DECOMPRESS
- ROWCOUNT_BIG
- ERROR_LINE
- SESSION_CONTEXT
- ERROR_MESSAGE
- SESSION_ID
- ERROR_NUMBER
- XACT_STATE



ⓘ Note

- The names of some Transact-SQL system functions begin with two *at* signs (@@). Although in earlier versions of SQL Server, the @@functions are referred to as global variables, @@functions aren't variables, and they don't have the same behaviors as variables. The @@functions are system functions, and their syntax usage follows the rules for functions.
- You can't use variables in a view.
- Changes to variables aren't affected by the rollback of a transaction.

Displaying global Variables

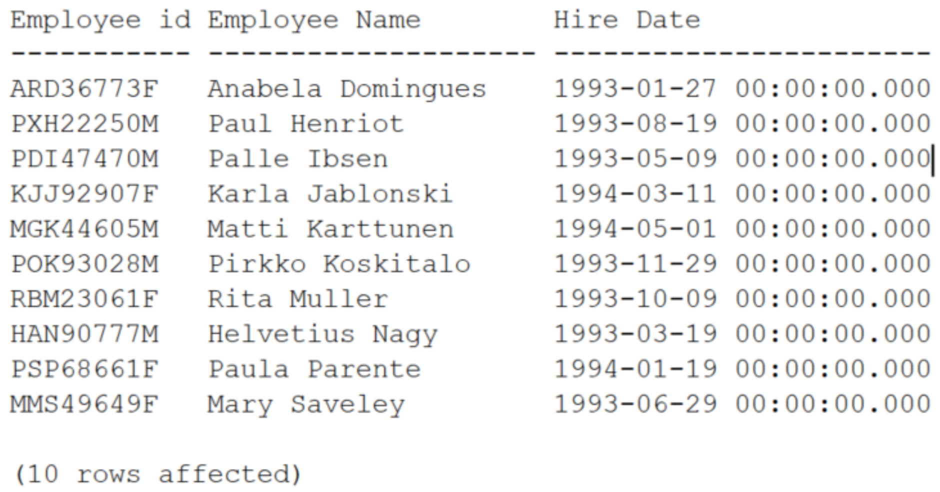


```sql
select @@LANGUAGE as Language
```

```
Language
-------------
us_english

(1 row affected)


Completion time: 2021-01-19T06:56:36.2788481-07:00
```

Exercise 3

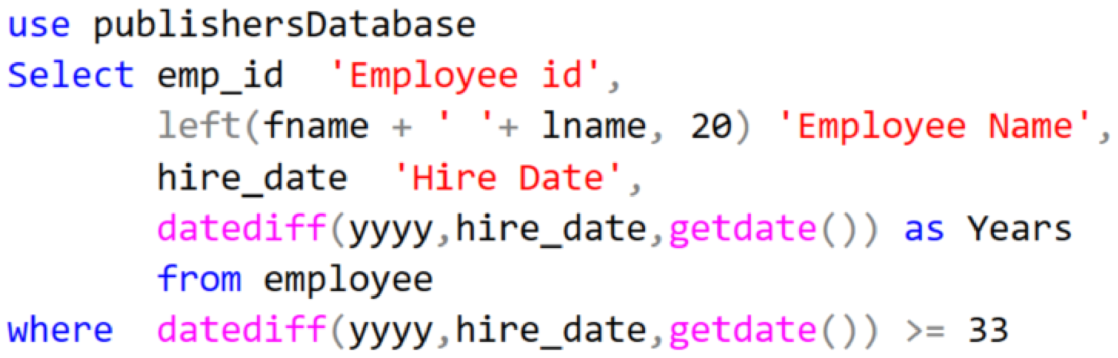
- Use the database publishersdatabase. Write a Select statement to display the employee id, first name + last name and the Hiredate for all employees who have been hired as from 1993 (from the employee table).
- Note the first name should be separated from the last name by a blank and the whole name should be limited to 20 characters.



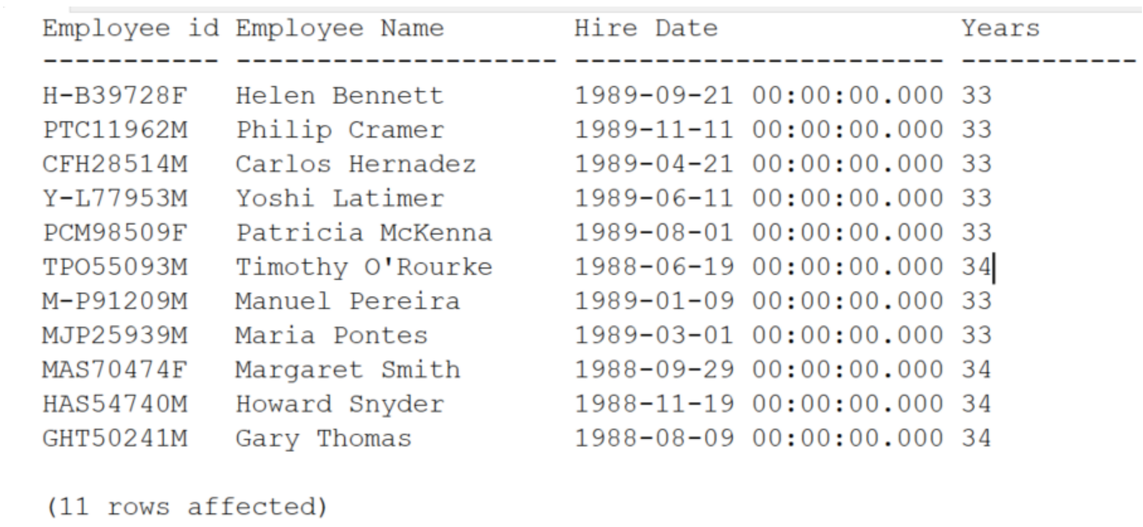
```
Employee id Employee Name        Hire Date
----------- -------------------- -----------------------
ARD36773F   Anabela Domingues    1993-01-27 00:00:00.000
PXH22250M   Paul Henriot         1993-08-19 00:00:00.000
PDI47470M   Palle Ibsen          1993-05-09 00:00:00.000
KJJ92907F   Karla Jablonski      1994-03-11 00:00:00.000
MGK44605M   Matti Karttunen      1994-05-01 00:00:00.000
POK93028M   Pirkko Koskitalo     1993-11-29 00:00:00.000
RBM23061F   Rita Muller          1993-10-09 00:00:00.000
HAN90777M   Helvetius Nagy       1993-03-19 00:00:00.000
PSP68661F   Paula Parente        1994-01-19 00:00:00.000
MMS49649F   Mary Saveley         1993-06-29 00:00:00.000

(10 rows affected)
```

Using calculated values for filtering

- Let’s say we want to display the employee id, employee name and hire date and number of Years for all employees who have been employed for at least 33 years from publishersdatabase.



```
use publishersDatabase
Select emp_id  'Employee id',
       left(fname + ' '+ lname, 20) 'Employee Name',
       hire_date  'Hire Date',
       datediff(yyyy,hire_date,getdate()) as Years
       from employee
where  datediff(yyyy,hire_date,getdate()) >= 33
```



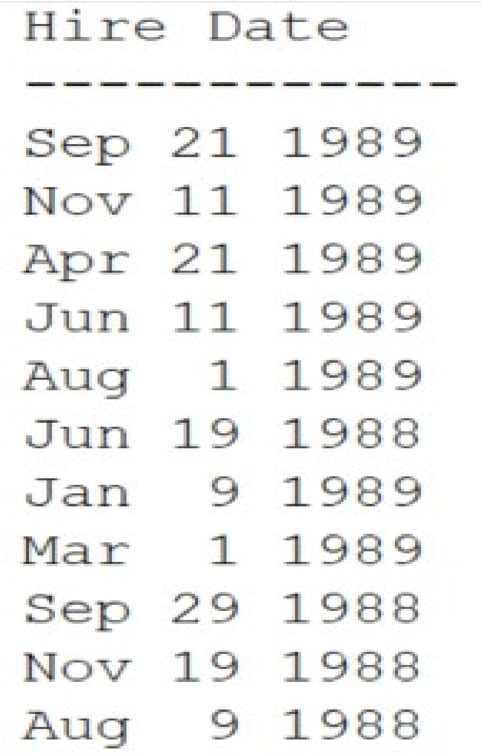
```
Employee id Employee Name        Hire Date               Years
----------- -------------------- ----------------------- -----------
H-B39728F   Helen Bennett        1989-09-21 00:00:00.000 33
PTC11962M   Philip Cramer        1989-11-11 00:00:00.000 33
CFH28514M   Carlos Hernadez      1989-04-21 00:00:00.000 33
Y-L77953M   Yoshi Latimer        1989-06-11 00:00:00.000 33
PCM98509F   Patricia McKenna     1989-08-01 00:00:00.000 33
TPO55093M   Timothy O'Rourke     1988-06-19 00:00:00.000 34
M-P91209M   Manuel Pereira       1989-01-09 00:00:00.000 33
MJP25939M   Maria Pontes         1989-03-01 00:00:00.000 33
MAS70474F   Margaret Smith       1988-09-29 00:00:00.000 34
HAS54740M   Howard Snyder        1988-11-19 00:00:00.000 34
GHT50241M   Gary Thomas          1988-08-09 00:00:00.000 34

(11 rows affected)
```

Formatting the Date output

- In all the previous displays, we have had the time component also displayed with the hire date. This wasn’t necessary.
- We can have the date without the time by converting to varchar and with one of the different styles.
- **Note that in all the queries that follow, we are using the database publishersdatabase**

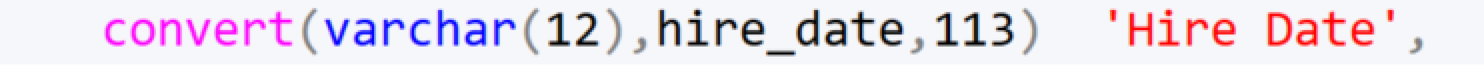
Formatting the Date output

```
…  convert(varchar(12),hire_date)  'Hire Date',  …
```

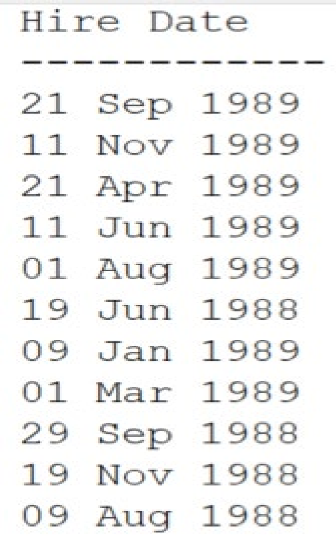


```
Hire Date
------------
Sep 21 1989
Nov 11 1989
Apr 21 1989
Jun 11 1989
Aug  1 1989
Jun 19 1988
Jan  9 1989
Mar  1 1989
Sep 29 1988
Nov 19 1988
Aug  9 1988
```

Formatting the Date output



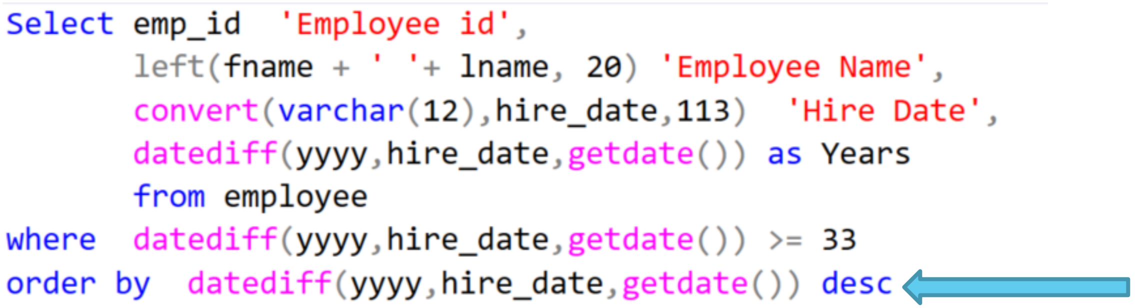
```
    convert(varchar(12),hire_date,113)  'Hire Date',
```



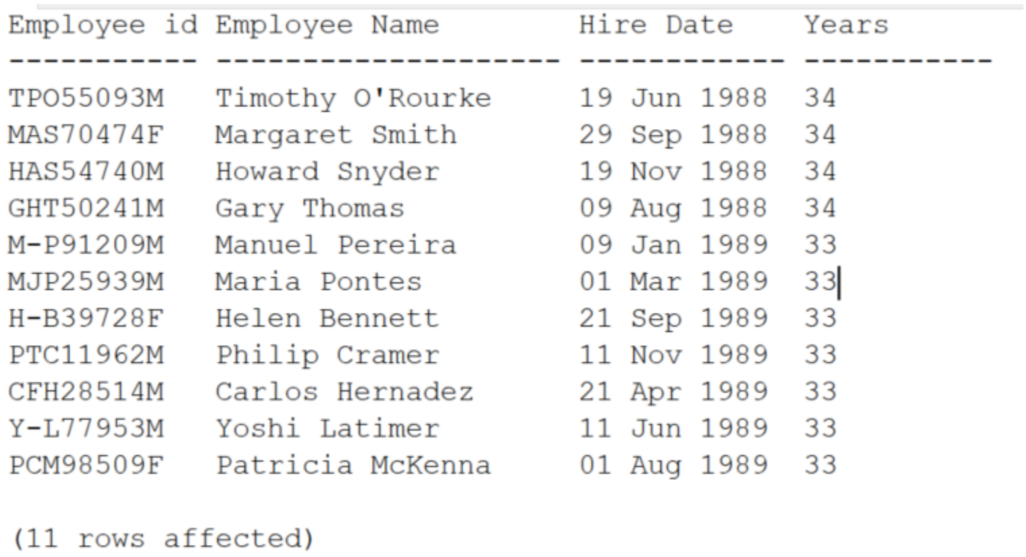
```
Hire Date
------------
21 Sep 1989
11 Nov 1989
21 Apr 1989
11 Jun 1989
01 Aug 1989
19 Jun 1988
09 Jan 1989
01 Mar 1989
29 Sep 1988
19 Nov 1988
09 Aug 1988
```

Using Calculated Values for ordering

We can also order by a calculated column



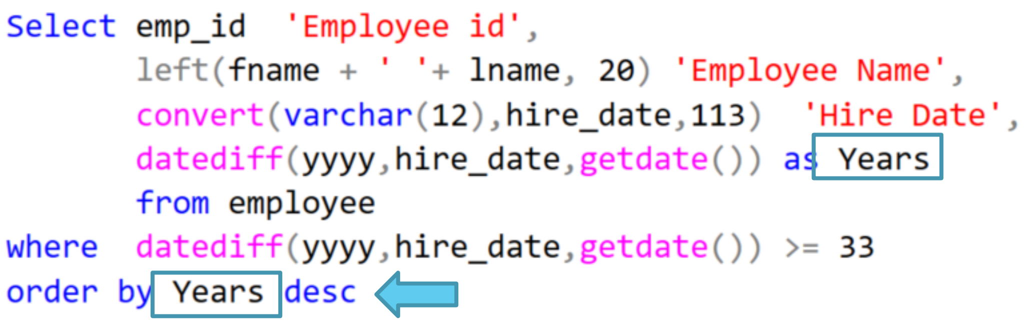
```
Select emp_id  'Employee id',
       left(fname + ' '+ lname, 20) 'Employee Name',
       convert(varchar(12),hire_date,113)  'Hire Date',
       datediff(yyyy,hire_date,getdate()) as Years
       from employee
where  datediff(yyyy,hire_date,getdate()) >= 33
order by  datediff(yyyy,hire_date,getdate()) desc
```



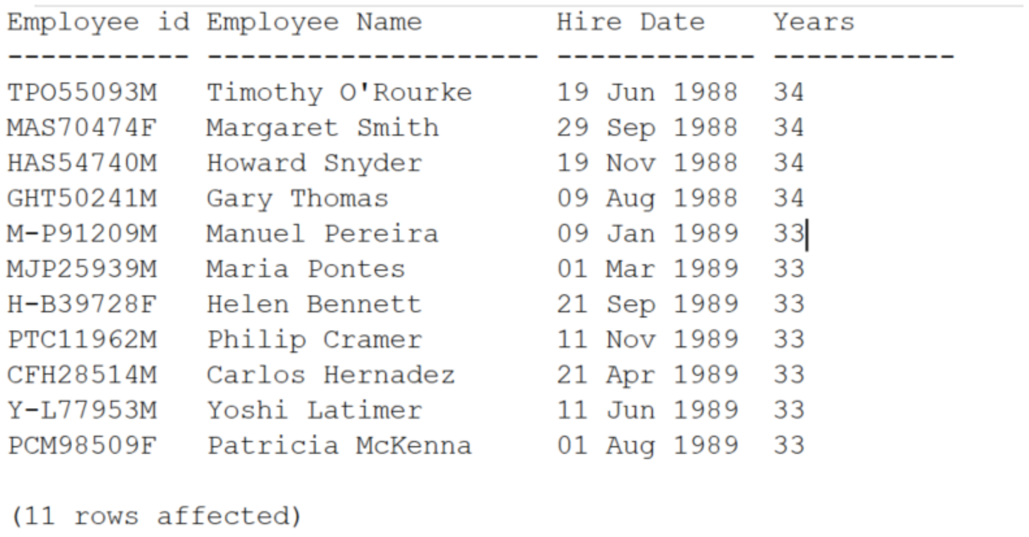
```
Employee id Employee Name        Hire Date    Years
----------- -------------------- ------------ -----------
TPO55093M   Timothy O'Rourke     19 Jun 1988  34
MAS70474F   Margaret Smith       29 Sep 1988  34
HAS54740M   Howard Snyder        19 Nov 1988  34
GHT50241M   Gary Thomas          09 Aug 1988  34
M-P91209M   Manuel Pereira       09 Jan 1989  33
MJP25939M   Maria Pontes         01 Mar 1989  33
H-B39728F   Helen Bennett        21 Sep 1989  33
PTC11962M   Philip Cramer        11 Nov 1989  33
CFH28514M   Carlos Hernadez      21 Apr 1989  33
Y-L77953M   Yoshi Latimer        11 Jun 1989  33
PCM98509F   Patricia McKenna     01 Aug 1989  33

(11 rows affected)
```

Using alias in order by clause

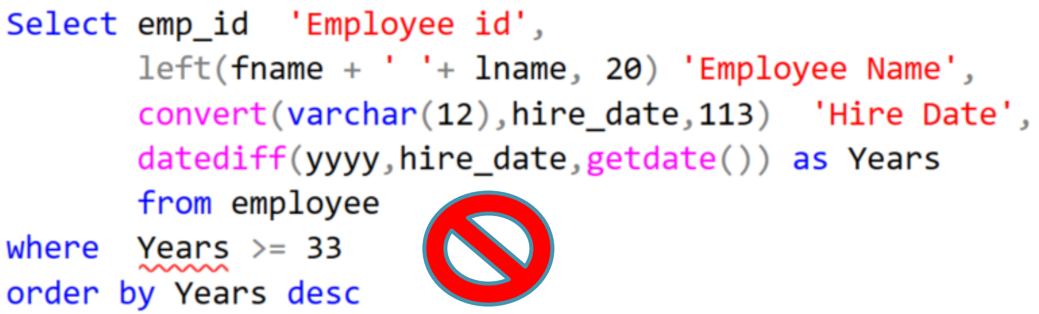
Interestingly, we can use an alias in an order by clause



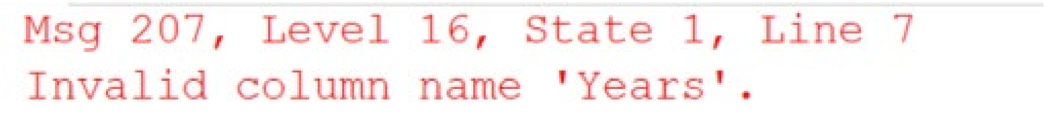
```
Select emp_id  'Employee id',
       left(fname + ' '+ lname, 20) 'Employee Name',
       convert(varchar(12),hire_date,113)  'Hire Date',
       datediff(yyyy,hire_date,getdate()) as Years
       from employee
where  datediff(yyyy,hire_date,getdate()) >= 33
order by Years desc
```



```
Employee id Employee Name         Hire Date    Years
----------- --------------------- ------------ -----------
TPO55093M   Timothy O'Rourke      19 Jun 1988  34
MAS70474F   Margaret Smith        29 Sep 1988  34
HAS54740M   Howard Snyder         19 Nov 1988  34
GHT50241M   Gary Thomas           09 Aug 1988  34
M-P91209M   Manuel Pereira        09 Jan 1989  33
MJP25939M   Maria Pontes          01 Mar 1989  33
H-B39728F   Helen Bennett         21 Sep 1989  33
PTC11962M   Philip Cramer         11 Nov 1989  33
CFH28514M   Carlos Hernadez       21 Apr 1989  33
Y-L77953M   Yoshi Latimer         11 Jun 1989  33
PCM98509F   Patricia McKenna      01 Aug 1989  33

(11 rows affected)
```



```
Select emp_id  'Employee id',
       left(fname + ' '+ lname, 20) 'Employee Name',
       convert(varchar(12),hire_date,113)  'Hire Date',
       datediff(yyyy,hire_date,getdate()) as Years
       from employee
where  Years >= 33
order by Years desc
```



```
Msg 207, Level 16, State 1, Line 7
Invalid column name 'Years'.
```

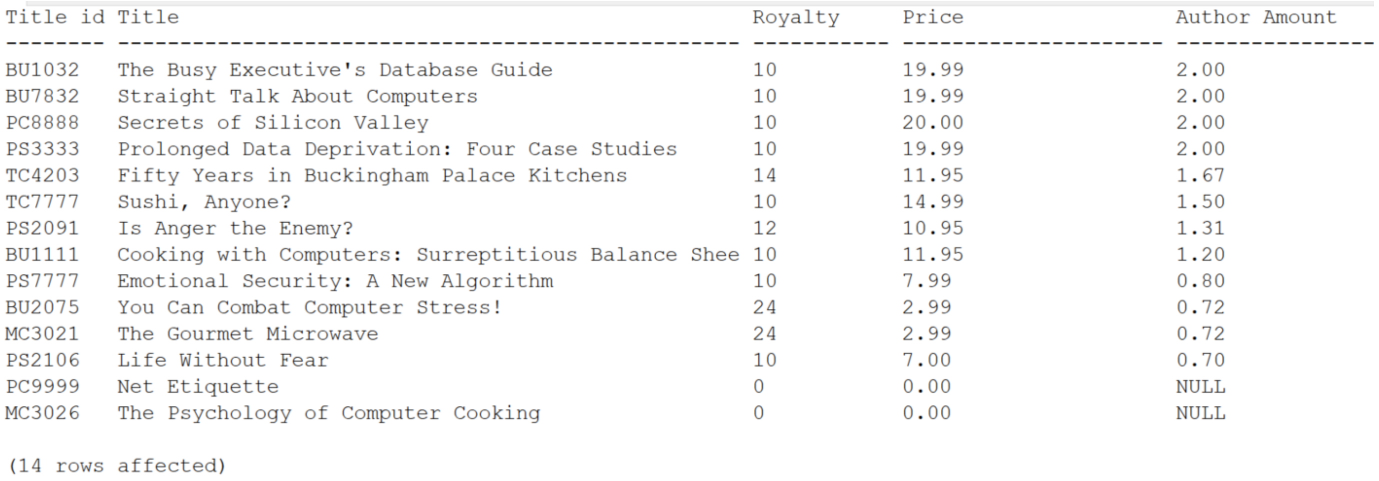
Can we use an alias in a where clause?

Note that while an alias can be used in an “order by” clause, it cannot be used in a “where” clause

Exercise 4

- Use the Publishers database
- Write a query that displays the title id, title, royalty, price and the amount the authors will obtain per copy (that will be royalty/100 * price) for all titles from the titles table where the amount obtained by the authors does not exceed $2.00. Include any rows where the amount obtained is null. When Royalty or price is null, display them as 0.
- Limit the titles to the first 50 characters.
- Answer on next slide

Exercise 4- Expected Output



```
Title id Title                                              Royalty     Price                 Author Amount
-------- -------------------------------------------------- ----------- --------------------- ----------------
BU1032   The Busy Executive's Database Guide                10          19.99                 2.00
BU7832   Straight Talk About Computers                      10          19.99                 2.00
PC8888   Secrets of Silicon Valley                          10          20.00                 2.00
PS3333   Prolonged Data Deprivation: Four Case Studies      10          19.99                 2.00
TC4203   Fifty Years in Buckingham Palace Kitchens          14          11.95                 1.67
TC7777   Sushi, Anyone?                                     10          14.99                 1.50
PS2091   Is Anger the Enemy?                                12          10.95                 1.31
BU1111   Cooking with Computers: Surreptitious Balance Shee 10          11.95                 1.20
PS7777   Emotional Security: A New Algorithm                10          7.99                  0.80
BU2075   You Can Combat Computer Stress!                    24          2.99                  0.72
MC3021   The Gourmet Microwave                              24          2.99                  0.72
PS2106   Life Without Fear                                  10          7.00                  0.70
PC9999   Net Etiquette                                      0           0.00                  NULL
MC3026   The Psychology of Computer Cooking                 0           0.00                  NULL

(14 rows affected)
```

Exercise 5

- Modify the previous query so that where the royalty or the price is null, they appear as ‘N/A’ instead of 0.