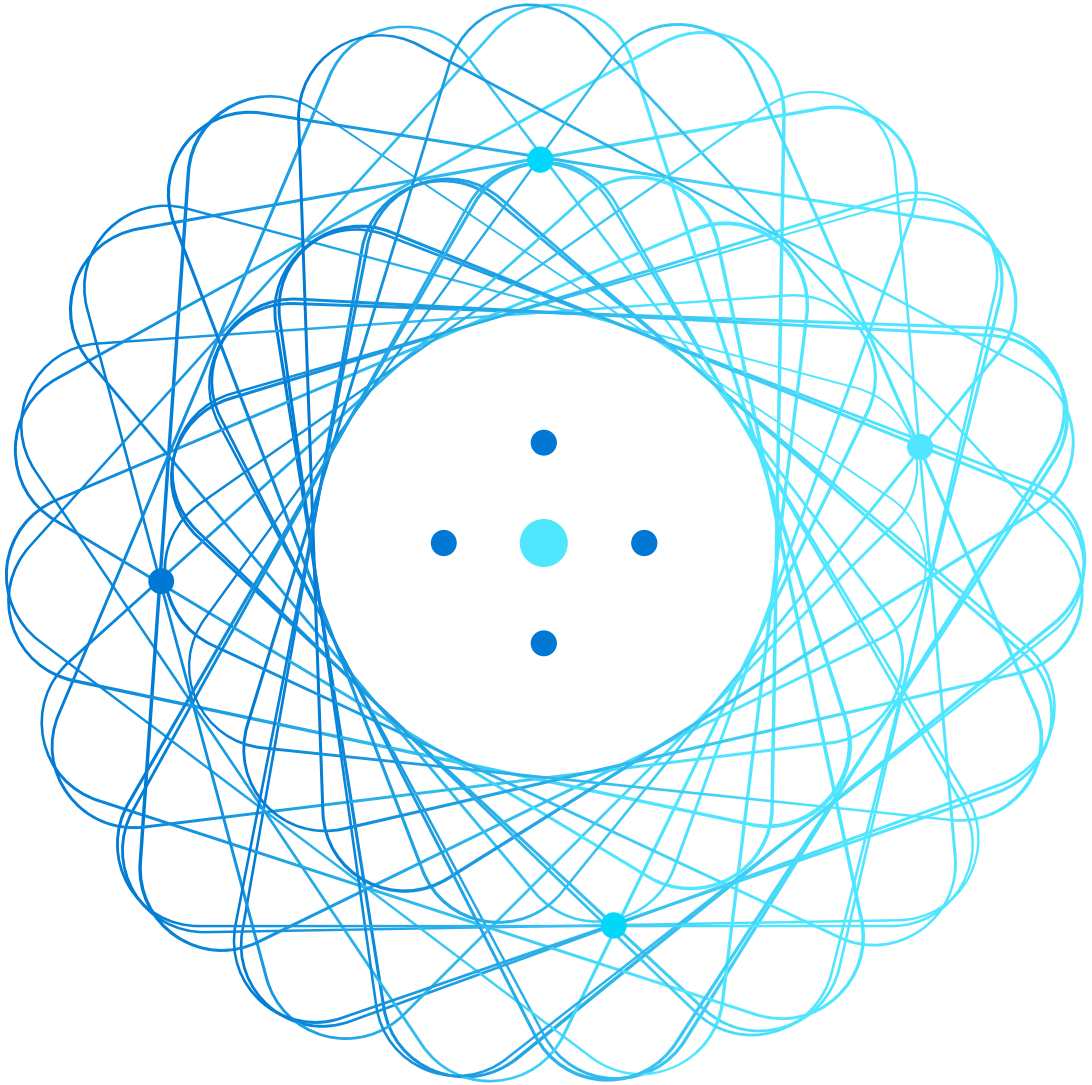
Microsoft Azure

DATAPOT

Module 2: Filtering Query Results and Data type



Module Agenda

Filtering Query Results

Data types in SQL Server

SQL data type functions

Lesson 1: Filtering Query Results

Where clause

WHERE Clause: used to filter records (extract only those records that fulfill a specified condition).

Syntax:

```
SELECT column1, column2, column3
FROM table_name
WHERE [condition]
```

Example:

Write a query using a WHERE clause that displays all the employees listed in the DimEmployee table who have the job title “Research and Development Engineer”. Display the EmployeeKey, the login ID, and the title for each one.

```
SELECT EmployeeKey, LoginID, Title
FROM DimEmployee
WHERE Title = 'Research and Development Engineer'
```

Operators with WHERE

Group	Operator	Description
Comparison operators	=	Equal
	<> Or !=	Not equal
	>	Greater than
	<	Less than
	>=	Greater than or equal
	<=	Less than or equal
Logical operators	AND	Return records that meet all the conditions separated by AND in WHERE clause.
	OR	Return records that meet any of the conditions separated by OR in WHERE clause.
	NOT	Return records that do not satisfy any of the conditions in WHERE clause.
SQL operators	[NOT] BETWEEN	Returns values [NOT] within a given range
	[NOT] IN	Specify multiple values in a WHERE clause (a shorthand for multiple OR conditions)
	IS [NOT] NULL	Returns records having [NOT] NULL values in the given fileds.
	[NOT] LIKE	Returns records that [DO NOT] match a specified pattern in a column.

SQL operators

BETWEEN Syntax (https://docs.microsoft.com/en-us/sql/t-sql/language-elements/between-transact-sql?view=sql-server-ver15)

The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.

The BETWEEN operator is inclusive: begin and end values are included.

```
SELECT column_name(s)

FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

IN Syntax (https://docs.microsoft.com/en-us/sql/t-sql/language-elements/in-transact-sql?view=sql-server-ver15)

```
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);
```

SQL operators

LIKE Syntax: https://docs.microsoft.com/en-us/sql/t-sql/language-elements/like-transact-sql?view=sql-server-ver15

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

Wildcard character	Description	Example
%	Any string of zero or more characters.	WHERE title LIKE '%computer%' finds all book titles with the word 'computer' anywhere in the book title.
_ (underscore)	Any single character.	WHERE fname LIKE '_ean' finds all four-letter first names that end with ean (Dean, Sean, and so on).
[ ]	Any single character within the specified range ([a-f]) or set ([abcdef]).	WHERE lname LIKE '[C-P]arsen' finds author last names ending with arsen and starting with any single character between C and P, for example Carsen, Larsen, Karsen, and so on. In range searches, the characters included in the range may vary depending on the sorting rules of the collation.
[^]	Any single character not within the specified range ([^a-f]) or set ([^abcdef]).	WHERE name LIKE 'de[^l]%' finds all author last names starting with de and where

Where clause – Exercise

Exercise: Use AdventureWorksDW2019

From table ***"DimEmployee"*** select all records that satisfy one of the following conditions:

- ***DepartmentName*** is equal to "Tool Design"
- ***Status*** does NOT include the value **NULL**
- ***StartDate*** in the period from '2009-01-01' to '2009-12-31'

And must have ***VacationHours*** >10

Lesson 2: Data type in SQL Server

Data Types- String types

Data type	Description	Max size
char(n)	Fixed width character string	8,000 characters
varchar(n)	Variable width character string	8,000 characters
varchar(max)	Variable width character string	1,073,741,824 characters
text	Variable width character string	2GB of text data
nchar(n)	Fixed width Unicode string	4,000 characters
nvarchar(n)	Variable width Unicode string	4,000 characters
nvarchar(max)	Variable width Unicode string	536,870,912 characters
ntext	Variable width Unicode string	2GB of text data
binary(n)	Fixed width binary string	8,000 bytes
Varbinary(n)	Variable width binary string	8,000 bytes
varbinary(max)	Variable width binary string	2GB
image	Variable width binary string	2GB

Data Types- Number types

Data type	Description	Max size
bit	Integer that can be 0, 1, or NULL	
tinyint	Allows whole numbers from 0 to 255	1 byte
smallint	Allows whole numbers between -2^{15} and 2^{15}	2 bytes
int	Allows whole numbers between -2^{31} and 2^{31}	4 bytes
bigint	Allows whole numbers between -2^{63} and 2^{63}	8 bytes
decimal (p,s)	Fixed precision and scale numbers. Allows numbers from -10^{38} +1 to 10^{38} -1.	5-17 bytes
numeric(p,s)	Fixed precision and scale numbers. Allows numbers from -10^{38} +1 to 10^{38} -1.	5-17 bytes
money	Monetary data from -2^{63} to 2^{63}	8 bytes
smallmoney	Monetary data from -214,748.3648 to 214,748.3647	4 bytes
float(n)	Floating precision number data from -1.79E + 308 to 1.79E + 308.	4 or 8 bytes
real	Floating precision number data from -3.40E + 38 to 3.40E + 38	4 bytes

Data Types- Datetime types

Data type	Description	Max size
datetime	From 1/1/1753 to 31/12/9999 with an accuracy of 3.33 milliseconds	8 bytes
datetime2	From 1/1/0001 to 31/12/9999 with an accuracy of 100 nanoseconds	6-8 bytes
smalldatetime	From 1/1/1900 to 06/06/2079 with an accuracy of 1 minute	4 bytes
date	Store a date only. From 1/1/0001 to 31/12/9999	3 bytes
time	Store a time only to an accuracy of 100 nanoseconds	3-5 bytes
datetimeoffset	The same as datetime2 with the addition of a time zone offset	8-10 bytes

NULL Values

NULL represents a *missing* or *unknown* value

ANSI behaviour for NULL values:

- The result of any expression containing a NULL value is NULL

```
2 + NULL = NULL

'MyString: ' + NULL = NULL
```

- Equality comparisons (=) always return false for NULL values, use IS NULL

`NULL = NULL` returns false

`NULL IS NULL` returns true

Useful functions:

`ISNULL(`*`column/variable, value`*`)`: Returns *value* if the column or variable is NULL

`* COALESCE(`*`column/variable1, column/variable2, ...`*`)`: Returns the value of the first non-NULL column or variable in the list

Conversion function

- Compatible data types can be implicitly converted
- Explicit conversion requires an explicit conversion function:
  CAST / TRY_CAST
  CONVERT / TRY_CONVERT
  STR

- **CAST() converts a value (of any type) into a specified datatype**

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

```
SELECT CAST('2022' as int) as new_datatype
```

- **CONVERT() converts a value (of any type) into a specified datatype**

```
-- CONVERT(datatype(length), expression, style)
```

```
SELECT CONVERT(int, '2022') as new_datatype
SELECT CONVERT(VARCHAR,GETDATE(),21) /* 22? */
```

(More Date and Time style: https://www.mssqltips.com/sqlservertip/1145/date-and-time-conversions-using-sql-server/)

Lesson 3: SQL data type functions

Date Function

- **DAY()** returns the day of the month for a given date

```
-- DAY(date) --
SELECT DAY('2022/03/10') AS get_day
```

- **MONTH()** returns the month part for a given date (a number from 1 to 12)

```
-- MONTH(date) --
SELECT MONTH'2022/03/10') AS get_month
```

- **YEAR()** returns the year part for a given date

```
-- YEAR(date) --
SELECT YEAR('2022/03/10') AS get_year
```

- **DATEPART()** returns a specified part of a date (as integer)

```
-- DATEPART(interval, date) --
SELECT DATEPART(year, '2022/03/10') AS date_part_int
```

Date Function

- **CURRENT_TIMESTAMP** returns the current date and time

```
SELECT CURRENT_TIMESTAMP
```

- **DATEADD()** adds a time/date interval to a date then returns the date

```
-- DATEADD(interval, number, date) --
SELECT DATEADD(year, 1, '2022-03-10') as date_add
```

- **DATEDIFF()** returns the difference between two dates

```
-- DATEDIFF(interval, date, date) --
SELECT DATEDIFF(year, '2021-03-10', '2022-03-10') as diff
/* DAY ? */
```

- **DATEFROMPARTS()** returns a date from the specified parts (year, month, and day values)

```
-- DATEFROMPARTS(year, month, day) --
SELECT DATEFROMPARTS(2022, 03, 10) AS date_from_parts
```

String Function

- **CHARINDEX()** returns the position of a substring in a string

```
-- CHARINDEX(substring, string, [start_position]) --
SELECT CHARINDEX('p', 'Datapot') AS char_indexc
```

- **PATINDEX()** returns the position of a pattern in a string

```
-- PATINDEX(%pattern%, string) –
SELECT PATINDEX('%pot%', 'Datapot')
```

- **SUBSTRING()** extracts some characters from a string

```
-- SUBSTRING(string, start, number_of_chars) --
SELECT SUBSTRING('Datapot', 1, 3) AS extract_string
```

- **CONCAT()** adds two or more strings together

```
-- CONCAT(string1, string2, ....) --
SELECT CONCAT('Data','pot') AS chars_combined

-- Another method --
SELECT 'Data' + 'pot' AS chars_combined
```

String Function

- **LEFT()** extracts a number of characters from a string (starting from left)

```
-- LEFT(string, number_of_chars) --
SELECT LEFT('Datapot', 4) AS extract_string
```

- **RIGHT()** extracts a number of characters from a string (starting from right)

```
-- RIGHT(string, number_of_chars) --
SELECT RIGHT('Datapot', 4) AS extract_string
```

- **LTRIM() or RTRIM()** removes leading spaces from a string

```
SELECT LTRIM('     Datapot') AS Left_Trimmed
SELECT RTRIM('Datapot     ') AS Right_Trimmed
```

- **LEN()** returns the length of a string

```
SELECT LEN('Datapot') as char_length
```

String Function

- **REPLACE()** replaces all occurrences of a substring within a string, with a new substring

```
-- REPLACE(original_string, substring, substitution) --
SELECT REPLACE('Datapot', 'pot', 'top')
```

- **REPLICATE()** repeats a string a specified number of times

```
-- REPLICATE(string, times) --
SELECT REPLICATE('Datapot ', 3)
```

- **REVERSE()** reverses a string

```
SELECT REVERSE('Datapot')
```

- **STUFF()** deletes a part of a string and then inserts another part into the string, starting at a specified position

```
-- STUFF(original_string, start, length, substitution) --
SELECT STUFF('Datapot', 1, 4, 'Tea')
```

- **LOWER()** converts a string to lower-case

```
SELECT LOWER('Datapot')
```

- **UPPER()** converts a string to upper-case

```
SELECT UPPER('Datapot')
```