Note: The above picture is a reference from the following website. www.kdnuggets.com

Getting started

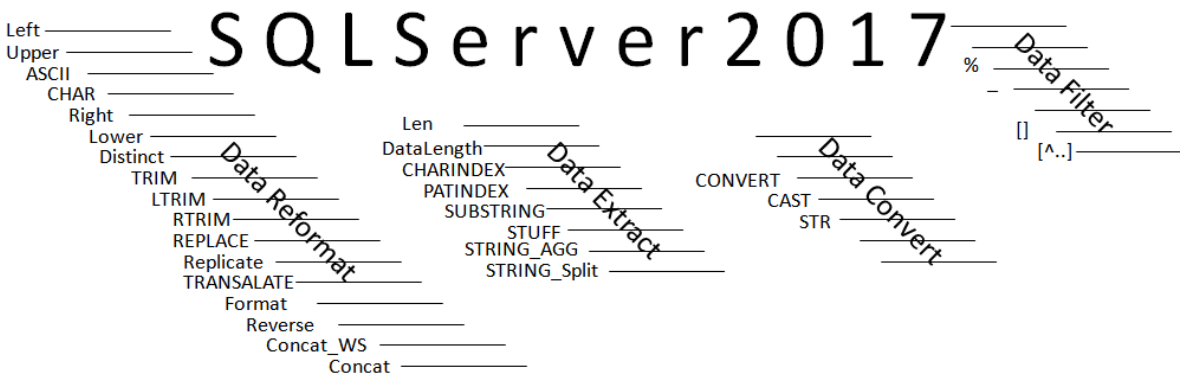
Let now take a deep dive into SQL string functions to see the different phases of data munging

SQL is further classified into Data Manipulation Language (DML) and Data Definition Language (DDL). These commands are used to work with data-sets more efficiently. We'll take a look at DML commands at later part of the article.

Now, let's discuss and analyze some of the SQL commands and understand why data-scientist needs to know these commands to do their work efficiently. In most cases, a majority portion of their work is about data gathering, data preparation, data cleaning, and data restructuring. After the data preparation phase, the scientist can move forward with the data analysis. In some scenarios, it's been assumed that about 70% to 80% of the time on the data science project is spent on data manipulation; if this is the case then most of that time is spent working with SQL queries.

The data cleansing is an art in data science; we often tend to collect data from multiple data sources. Many times the same data is stored differently in multiple systems. Let us classify the data munging process into the following categories:

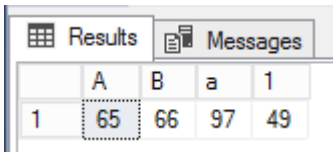
1. Data reformatting
2. Data extracting
3. Data filtering
4. Data converting

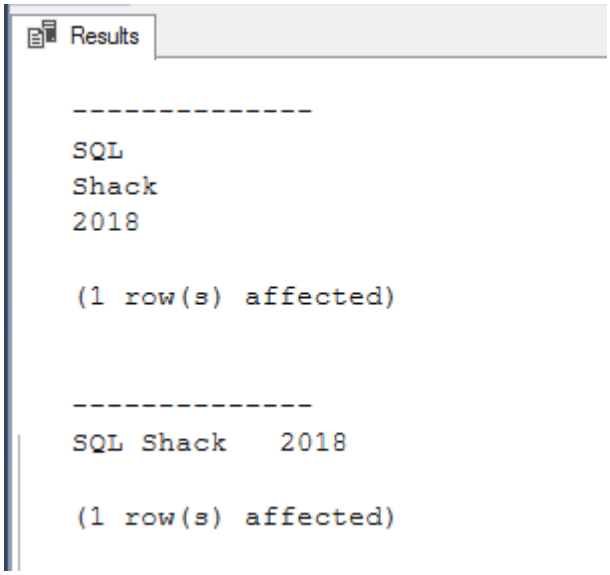


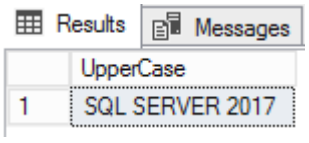
Data refactoring

As a basic principle, whenever we start working with a new data set, it is recommended to spend more time to understand the type and nature of the data.

For example, one couple of data-sets may use abbreviations for departments and in some other datasets it may spell out the full name. We need to reformat data to get it into a consistent format.

Character	Description	Example
ASCII	The **ASCII()** SQL String function servers as a characters encoding standard format.	In the following example, the ASCII values are returned for the given input `SELECT ASCII('A') A, ASCII('B') B, ASCII('a') AS a, ASCII('1') as '1'` 

CHAR	The **CHAR()** SQL string function converts an int ASCII code to a character value. Use CHAR to insert control characters into character strings.	The control character are:
		• Tab is char(9)
		• Line feed is char(10)
		• Carriage return is char(13)
		`select 'SQL'+Char(13)+'Shack'+Char(13)+'2018'`
		`select 'SQL'+Char(9)+'Shack'+Char(9)+'2018'`
		 

UPPER	The **UPPER()** SQL string function is used to convert the lower case to upper case of the given string	Convert the text 'SQL Server 2017' to upper-case `SELECT UPPER('SQL Server 2017')` 

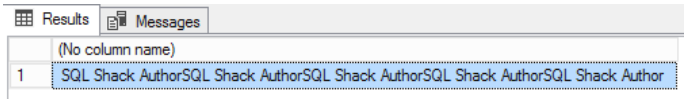
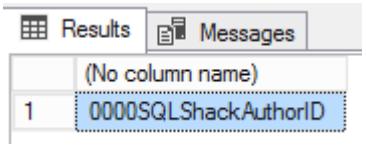
LOWER	The **LOWER()** SQL string function is used to convert the upper case to lower case of the given string	Convert the text 'SQL Server 2017' to lower-case `SELECT LOWER('SQL Server 2017')` Output: SQL Server 2017

CONCAT	The **CONCAT()** SQL string function is used to concatenate two or more string	Concatenate strings 'SQL Shack' and '2018' to SQL Shack 2018.
		`SELECT CONCAT('SQL Shack','2018')`
		`SELECT CONCAT('SQL Shack','2018' ,char(13),'2019',' ','2020')`
		Output: SQL Shack2018 2019 2020

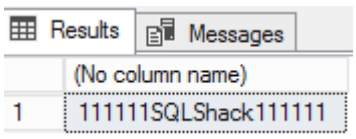
DISTINCT	The **DISTINCT** keyword is used to eliminates duplicate records from the SQL results	`SELECT DISTINCT [CustomerID]`
		`,[PersonID]`
		`,[StoreID]`
		`FROM [AdventureWorks2014].[Sales].[Customer]`

TRIM	The **TRIM()** SQL string function is used to remove blanks from the leading and trailing position of the given string	In the following example removes spaces from before and after the word SQL Server 2017. `SELECT TRIM('  SQL Server 2017  ')` Output: SQL Server 2017
LTRIM	**LTRIM()** SQL string function is used to remove the leading blanks from a given string	In the following example removes the leading spaces from the word SQL Server 2017 `SELECT LTRIM('  SQL Server 2017')` Output: SQL Server 2017
RTRIM	**RTRIM()** SQL string function is used to remove trailing blanks from a given string	In the following example removes the trailing spaces from the word SQL Server 2017 `SELECT RTRIM('SQL Server 2017  ')` Output: SQL Server 2017
RIGHT	The **RIGHT()** SQL string function is used to return a specified number of characters from the right side of the given string	The following example returns the 4 rightmost characters of the word SQL Server 2017. `SELECT RIGHT ('SQL Server 2017', 4)` Output: 2017
LEFT	The **LEFT()** SQL string function is used to return a specified number of characters from the	The following example returns the 10 leftmost characters of the word SQL Server 2017.

	number of characters from the left side of the given string	`SELECT LEFT ('SQL Server 2017', 10)` Output: SQL Server
REPLACE	The **REPLACE()** SQL string function used to replace all the occurrences of a source string with a target string of a given string	Replaces the string 'vNext' with '2018'. `SELECT REPLACE('SQL Server vNext','vNext','2017')` Output: SQL Server 2017

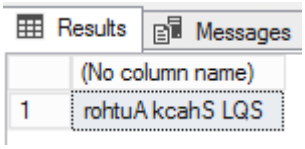
REPLICATE	The **REPLICATE()** SQL string function is used repeat the given string into the specified number of times	The following example, the string 'SQL Shack The string 'SQL Shack Author' is repeated 5 times
		`SELECT REPLICATE('SQL Shack Author', 5)`
		 
		The following example replicates a 0 character four times in front of a text
		`SELECT REPLICATE('0',4)+'SQLShackAuthorID'`
		 

SPACE	The **SPACE()** SQL string function replicates the number blanks that we want add to the given string	The following example concatenates 'SQL Shack', two spaces and the word '2018' `SELECT 'SQL Shack'+SPACE(2)+'2018'` Output: SQL Shack 2018

TRANSLATE	The **TRANSLATE()** SQL string function is used to perform a one-to-one, single-character substitution of a given string	In the following example the string replacement is performed using translate function.
		`DECLARE @Str NVARCHAR(MAX)`
		`SET @Str = '{~~[##SQLShack##]~~}'`
		`SELECT TRANSLATE(@Str, '#[]{}~~', '1111111');`
		 
		You can find more information here

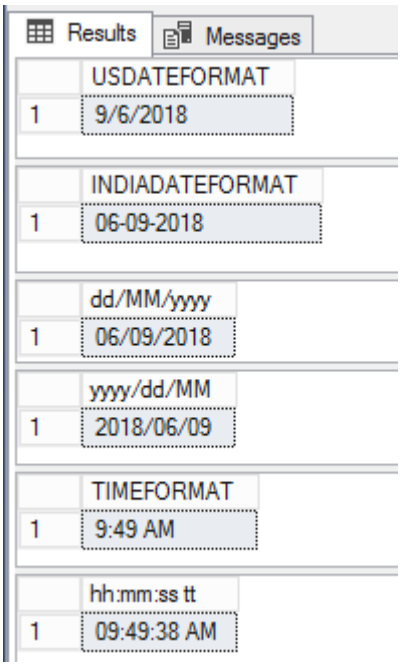
REVERSE	The **REVERSE()** SQL string function is used to get a mirror image of the given string	The following example returns the word with the characters reversed. `SELECT REVERSE('SQL Shack Author')`

Output:



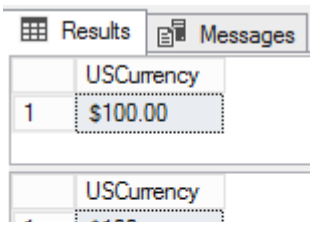
FORMAT	The **FORMAT()** SQL string function is used to return specified formatted value	The FORMAT SQL string function introduced in SQL SERVER 2012. It returns the value to format in by specified format and optional culture in SQL Server 2017. Examples for Date and Time formats

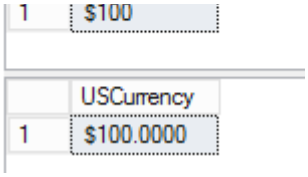
```
SELECT FORMAT(GETDATE(),'d','en-US') [USDATEFORMAT]
SELECT FORMAT(GETDATE(),'d','en-IN') [INDIADATEFORMAT]
SELECT FORMAT(GETDATE(),'dd/MM/yyyy') [dd/MM/yyyy]
SELECT FORMAT(GETDATE(),'yyyy/dd/MM') [yyyy/dd/MM]
SELECT FORMAT(GETDATE(),'t') AS [TIMEFORMAT]

SELECT FORMAT(GETDATE(),'hh:mm:ss tt') [hh:mm:ss tt]
```



Examples for Currency

```
SELECT FORMAT(100,'C','en-US')AS [USCurrency]

SELECT FORMAT(100,'C0','en-US')AS [USCurrencyNoDecimal]
SELECT FORMAT(100,'C4','en-US')AS [USCurrencyWith4Decimal]
```

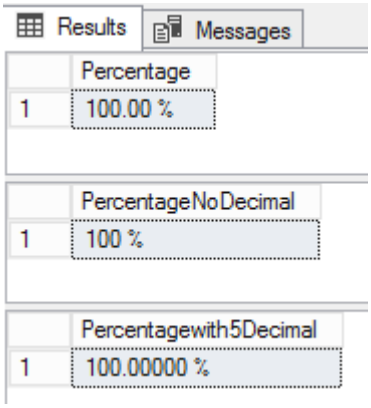




	USCurrency
1	$100.0000

Example for percentage

```
SELECT FORMAT(1,'P','en-US')AS [Percentage]
SELECT FORMAT(1,'P0','en-US')AS [PercentageNo
Decimal]
SELECT FORMAT(1,'P5','en-US')AS [Percentagewi
th5Decimal]
```



Results Messages

	Percentage
1	100.00 %

	PercentageNoDecimal
1	100 %

	Percentagewith5Decimal
1	100.00000 %

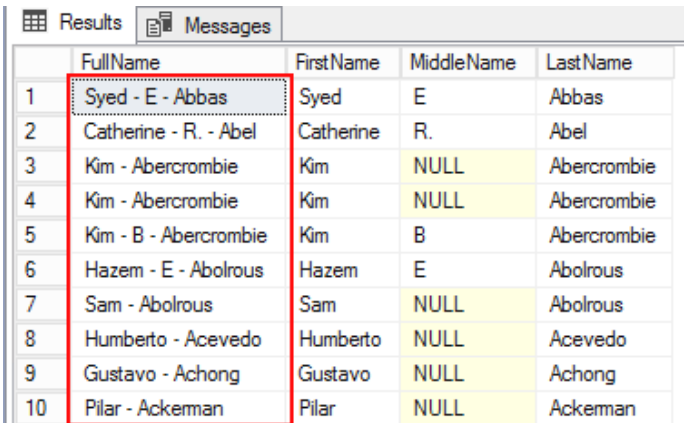
CONCAT_WS

The **CONCAT_WS()** SQL string function is a Concatenate with Separator and is a special form of CONCAT()

Concat_WS() emulates the behavior of stuff and Coalesce function. In the following example, '-' is the delimiter specified in the first argument followed by firstname, Middle name and lastname. The output concatenates three columns from the Person table separating the value with a '-'

```
USE Adventureworks2014
GO
SELECT TOP 10
    CONCAT_WS
    (
        ' - ' ,
        FirstName ,
        MiddleName ,
        LastName
    ) as FullName,
    FirstName, MiddleName,LastName
FROM [Person].[Person]
```

Output:



Results Messages

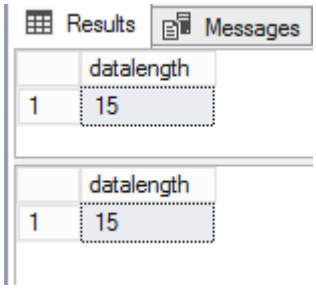
	FullName	FirstName	MiddleName	LastName
1	Syed - E - Abbas	Syed	E	Abbas
2	Catherine - R. - Abel	Catherine	R.	Abel
3	Kim - Abercrombie	Kim	NULL	Abercrombie
4	Kim - Abercrombie	Kim	NULL	Abercrombie
5	Kim - B - Abercrombie	Kim	B	Abercrombie
6	Hazem - E - Abolrous	Hazem	E	Abolrous
7	Sam - Abolrous	Sam	NULL	Abolrous
8	Humberto - Acevedo	Humberto	NULL	Acevedo
9	Gustavo - Achong	Gustavo	NULL	Achong
10	Pilar - Ackerman	Pilar	NULL	Ackerman

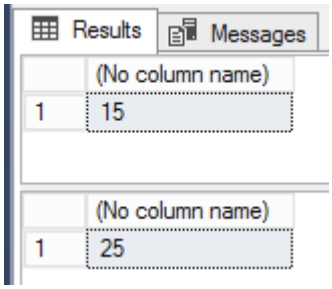
You can find more information here

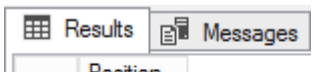
Data Extracting

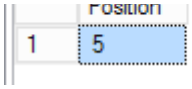
In addition to matching and reformatting strings, we sometimes need to take them apart and extract pieces of stings. SQL Server provides some general purpose SQL string functions for extracting and overriding strings. Let's start with a simple string that's easy to experiment

Character	Description	Usage

LEN	The **LEN()** SQL string function is used to determine the length of the given string excluding the trailing blanks	The following example selects the number of characters with an exclusion of the trailing spaces.
		`SELECT LEN('SQL Server 2017')`
		`SELECT LEN('SQL Server 201 7 ')`
		 

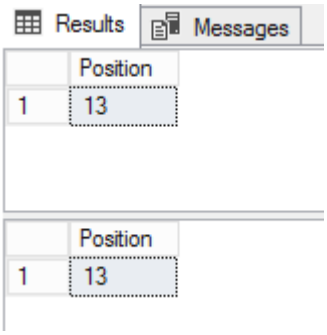
DATALENGTH	The **DATALENGTH()** SQL string function excludes the trailing blanks in a given string. If this is a problem, then use DATALENGTH SQL string function which includes the trailing blanks.	In this example, the trailing blanks are also considered while evaluating string length.
		`select DATALENGTH('SQL Server 2017' )`
		`select DATALENGTH('SQL Server 201 7 ')`
		 

CHARINDEX	The **CHARINDEX()** SQL string function is used to return the location of a substring in a given string.	In the following example, the starting position 'Shack' of the first expression will be returned. `SELECT CHARINDEX('Shack', 'SQL Shack Author SQL Shack Author') AS Position` 



PATINDEX	The **PATINDEX()** SQL string function is used to get the starting position of the first occurrence of given pattern in a specified expression	In the following examples, '%' and '_' wildcard characters are used to fin the position of the pattern in a given expression. PATINDEX works just like LIKE operator but it returns the matching position.

```
SELECT PATINDEX('%thor%', 'SQL Shac
k author') AS Position;
SELECT PATINDEX('%t__r%', 'SQL Shac
k author') AS Position;
```

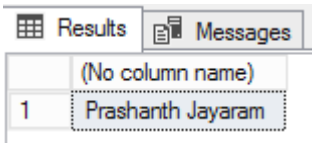


SUBSTRING	The **SUBSTRING()** SQL string function is used to returns a specific portion of a given string	The following query returns only the part of an input character string. In this example,

```
SELECT SUBSTRING('SQL Shack Author
 Prashanth Jayaram', 17,18)
```

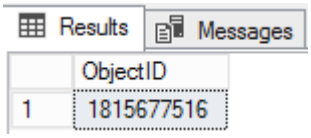
Example 1: In the below example, a part of given string "Prashanth Jayaram" is extracted dynamically using LEN and SQL Server SUBSTRING functions.

```
DECLARE @Str varchar(100)='SQL Shac
k Author Prashanth Jayaram'
SELECT SUBSTRING(@str, PATINDEX('%P
rashanth%',@str),Len(@str)-PATINDEX
('%Prashanth%',@str)+1)
```



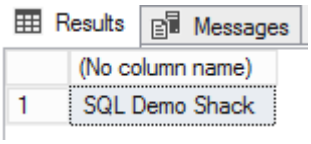
Example 2: In the example, we can see the extraction of ObjectID from the given string

```
DECLARE @str varchar(100)='HoBt 0:A
CQUIRE_LOCK_SCH_M OBJECT: 0:1815677
516:0'
SELECT Substring(@str,PATINDEX('%:
 0%', @str)+4,PATINDEX('%:0%', @str
)-PATINDEX('%: 0%', @str)-4) [Objec
tID]
```



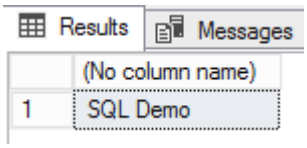
STUFF	The **STUFF()** SQL string function is used to place a string within another string	The following example returns a character string created by inserting a word Demo at the starting position 5 without deleting any letters from the given string 'SQL Shack'

```
SELECT STUFF('SQL  Shack ',5, 0, 'Demo')
```



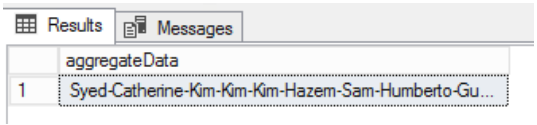
The following example returns a character string created by deleting 6 characters from the first string, SQL Shack, starting at position 5, at Shack, and inserting the second string 'Demo' at the deletion point.

```
SELECT STUFF('SQL Shack ',5, 6, 'Demo')
```



STRING_AGG	The **STRING_AGG()** SQL string function is the an aggregate function used to compute a single result from a set of input values	The following example returns the names separated by '-' in a single result set.

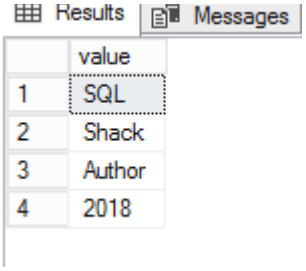
```
USE Adventureworks2014
GO
SELECT STRING_AGG (CAST(FirstName as VARCHAR(MAX)), '-') AS aggregateData
FROM Person.Person;
```



You can find more information here

STRING_SPLIT	The **STRING_SPLIT()** SQL string function is used to splits the input string by a specified separation character and returns the output split values in the form of table	SQL Server 2016 introduced a new STRING_SPLIT table-valued function. In an earlier version, we used to write function, CLR code to decode the values.

```
SELECT * from STRING_SPLIT('SQL,Shack,Author,2018',' ')
```



Data conversion

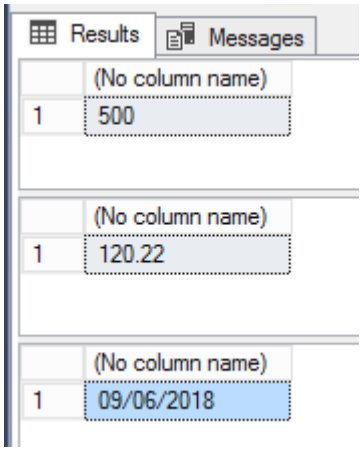
Sometimes we will need to reformat numbers. This is especially true when we use calculations that have results with large numbers of decimal digits

CONVERT

The **CONVERT()** SQL string function is used to convert an expression from one data type to another data type. The CONVERT SQL string function accepts in a style parameter which is used for formatting the SQL output

Implicit conversions do not require either the CAST function or the CONVERT function. Only explicit conversions require specification of the CAST or the CONVERT function. When converting a value from float or numeric to an integer, the CONVERT() SQL string function will truncate the result. For other conversions, the CONVERT() SQL string function will round the result.

```
SELECT CONVERT(INT, 500.55);
SELECT CONVERT(decimal(10,2),120.2245)
SELECT CONVERT(varchar, getdate(), 101);
```



You can find more information here

CAST

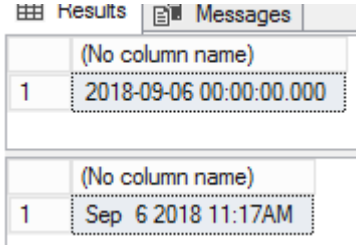
The **CAST()** SQL string function is used to convert an expression from one data type to another data type.

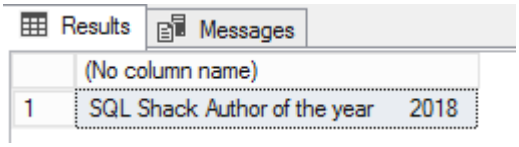
Convert an expression from valid date string to DateTime.

```
SELECT CAST('2018-09-06' AS datetime);
```

In the following example, the DateTime type is converted to another varchar type.

```
SELECT CAST(getdate() as varchar(20));
```



STR	The **STR()** SQL string function converts a numeric value into a string	The following example converts an integer to character string and concatenate the value with the first string. `select TRIM('SQL Shack Author of the year')+ STR(2018)` 

Data filtering

The use of the WHERE and HAVING clauses in a SELECT statement control the subset of the output from the source tables.

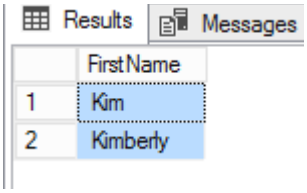
1. Compare search conditions using comparison operators
2. Range search using between, and, and or clause
3. List search using in operator and or clause

Regular expression search is based on patterns, wildcards, and special characters. Even if you never ever write CLR, regex (as it's also known) can be useful to you today, right now. Open a new query window in SQL Server Management Studio.

The regular expression transformation exposes the power of regular expression matching within the pipeline. One or more columns can be selected, and for each column an individual expression can be applied. The way multiple columns are handled can be set on the options page. The AND option means all columns must match, whilst the OR option means only one column has to match. If rows pass their tests then rows are passed down the successful match output. Rows that fail are directed down the alternate output

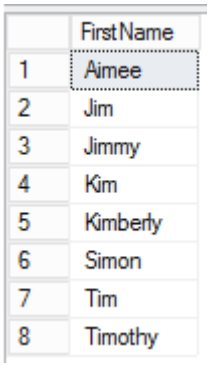
Character	Description	Usage

%	Matches a string of zero or more characters.	The following example returns the any values that matches the string 'Kim'
		`USE Adventureworks2014`
		`GO`
		`SELECT DISTINCT FirstName FROM Person.Person Where FirstName like '%Kim%'`



Underscore (_)	To Match a single character of given string	The following example returns the any values whose first characters is unknown followed by 'im'

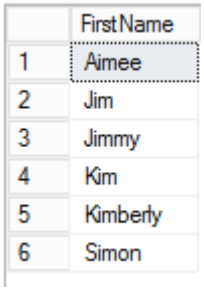
```
SELECT DISTINCT FirstName FROM Person.Person Where FirstName like '_im%'
```



	FirstName
1	Aimee
2	Jim
3	Jimmy
4	Kim
5	Kimberly
6	Simon
7	Tim
8	Timothy

[...]	Matches any character within a specified range	The following example returns the values whose first characters is in the range A to S followed by 'im'

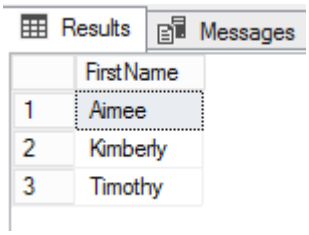
```
SELECT DISTINCT FirstName FROM Person.Person Where FirstName like '[A-S]im%'
```



	FirstName
1	Aimee
2	Jim
3	Jimmy
4	Kim
5	Kimberly
6	Simon

[^...]	Matches any character not within a specified range	The following example returns the values whose first characters are unknown; second-and-third character is 'im' and fourth character that matches not within l to Z range.

```
SELECT DISTINCT FirstName FROM Person.Person Where FirstName like '_im[^l-z]%'
```



Summary