**Spark built-in functions**

**Documentation:** [**Functions — PySpark 3.5.3 documentation**](https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/functions.html)

**Array Operations**

Functions designed to work with array columns.

* arrays\_zip(): Merges the values of the arrays into a struct.
* array(): Creates a new array column.
* array\_contains(): Returns true if the array contains a given value.
* array\_distinct(): Removes duplicate values from the array.
* array\_except(): Returns an array of the elements in the first array but not in the second array.
* array\_intersect(): Returns an array of the elements in both arrays.
* array\_join(): Concatenates the elements of an array using a delimiter.
* array\_max(): Returns the maximum value in the array.
* array\_min(): Returns the minimum value in the array.
* array\_position(): Returns the position of the first occurrence of an element in the array.
* array\_remove(): Removes all occurrences of a given value from the array.
* array\_repeat(): Returns a new array with repeated elements.
* array\_sort(): Sorts the array in ascending order.
* array\_union(): Returns an array of the elements in both arrays, without duplicates.
* explode(): Creates a new row for each element in the array.
* posexplode(): Like explode(), but includes the position of the element in the array.
* flatten(): Flattens an array of arrays into a single array.
* reverse(): Reverses the order of the elements in the array.
* size(): Returns the length of the array.
* slice(): Subsets the array starting from a specified position.

**Conditional Functions**

These functions are used to apply conditional logic within DataFrames.

* when(condition, value): Similar to SQL’s CASE WHEN, returns a value when a condition is true.
* otherwise(): Specifies the value to return if the when() conditions are not met.
* ifnull(): Returns the second value if the first is null, otherwise returns the first.
* nvl(): An alias of ifnull().
* nvl2(): Returns the second value if the first is not null; otherwise, it returns the third value.
* nullif(): Returns null if both values are equal, otherwise returns the first value.

**Map Operations**

Functions that operate on map columns.

* map(): Creates a new map column.
* map\_concat(): Concatenates multiple maps into one.
* map\_entries(): Converts a map into an array of structs with key and value fields.
* map\_from\_arrays(): Creates a map from two arrays (keys and values).
* map\_keys(): Returns an array of the keys in the map.
* map\_values(): Returns an array of the values in the map.
* element\_at(): Returns the value associated with the given key in the map.

**String Operations**

Functions for manipulating and working with string columns.

* concat(): Concatenates multiple columns or strings.
* concat\_ws(): Concatenates multiple columns or strings with a given separator.
* instr(): Returns the position of the first occurrence of a substring.
* length(): Returns the length of a string.
* lower(): Converts a string to lowercase.
* upper(): Converts a string to uppercase.
* regexp\_extract(): Extracts a substring using a regular expression.
* regexp\_replace(): Replaces substrings that match a regular expression.
* split(): Splits a string into an array based on a delimiter.
* substring(): Extracts a substring from a string.
* replace(): Replaces all occurrences of a substring with another substring.
* translate(): Replaces characters in a string with other characters.
* trim(): Trims the spaces from both ends of a string.
* ltrim(): Trims spaces from the left side of a string.
* rtrim(): Trims spaces from the right side of a string.
* initcap(): Capitalizes the first letter of each word.
* soundex(): Returns the Soundex code for a string.
* levenshtein(): Returns the Levenshtein distance between two strings.

**Math Operations**

Functions for performing mathematical operations on numeric columns.

* abs(): Returns the absolute value.
* ceil(): Returns the smallest integer greater than or equal to the value.
* floor(): Returns the largest integer less than or equal to the value.
* round(): Rounds a number to the nearest integer or specified decimal places.
* sqrt(): Returns the square root.
* log(): Returns the natural logarithm.
* log10(): Returns the base 10 logarithm.
* exp(): Returns the exponential value of a number.
* sin(), cos(), tan(): Trigonometric sine, cosine, and tangent.
* asin(), acos(), atan(): Inverse trigonometric functions.
* signum(): Returns the sign of a number (-1, 0, or 1).
* pow(): Raises a number to a given power.
* greatest(): Returns the greatest value among the arguments.
* least(): Returns the least value among the arguments.
* rand(): Generates a random number between 0 and 1.
* randn(): Generates a random number from the normal distribution.
* pi(): Returns the value of Pi.
* degrees(): Converts radians to degrees.
* radians(): Converts degrees to radians.

**Date and Time Operations**

Functions for working with date and timestamp columns.

* current\_date(): Returns the current date.
* current\_timestamp(): Returns the current timestamp.
* date\_add(): Adds a specified number of days to a date.
* date\_sub(): Subtracts a specified number of days from a date.
* datediff(): Returns the difference in days between two dates.
* add\_months(): Adds a specified number of months to a date.
* months\_between(): Returns the number of months between two dates.
* year(), month(), dayofmonth(): Extracts the year, month, day from a date.
* hour(), minute(), second(): Extracts the hour, minute, second from a timestamp.
* to\_date(): Converts a string to a date.
* to\_timestamp(): Converts a string to a timestamp.
* from\_unixtime(): Converts Unix time to a timestamp.
* unix\_timestamp(): Converts a timestamp to Unix time.
* date\_format(): Formats a date or timestamp as a string.
* last\_day(): Returns the last day of the month for a given date.
* next\_day(): Returns the first date after a given date that falls on the specified day of the week.

**Aggregate Functions**

Functions that aggregate data across rows.

* count(): Returns the count of rows.
* countDistinct(): Returns the count of distinct values.
* sum(): Returns the sum of values.
* avg(): Returns the average of values.
* max(): Returns the maximum value.
* min(): Returns the minimum value.
* stddev(): Returns the standard deviation.
* variance(): Returns the variance.
* first(): Returns the first value.
* last(): Returns the last value.
* collect\_list(): Returns a list of all values.
* collect\_set(): Returns a set of all distinct values.

**Advanced DataFrame Operations**

These are some of the more advanced functions that do not fit directly into other categories but are useful for certain types of data manipulation.

* **broadcast()**: Marks a DataFrame as small enough for broadcasting during join operations.
* **approx\_count\_distinct()**: Returns the approximate count of distinct items using the HyperLogLog algorithm.
* **cube()**: Computes aggregations on a multidimensional cube.
* **rollup()**: Similar to cube(), but provides hierarchical rollups (useful for subtotal calculations).
* **grouping():** Used to differentiate between aggregated and non-aggregated data when using cube or rollup.
* **pivot()**: Pivots a DataFrame by turning distinct values from one column into multiple columns.
* **to\_json()**: Converts a struct (or array of structs) to a JSON string.
* **from\_json()**: Parses a JSON string into a struct or array of structs.
* **schema\_of\_json()**: Infers the schema of a JSON string.
* **schema\_of\_csv()**: Infers the schema of a CSV string.
* **to\_csv()**: Converts a struct or array of structs into a CSV string.

**Hashing Functions**

Functions that generate hash values, often used for unique identifiers or partitioning.

* **hash()**: Returns a hash value of the column.
* **md5()**: Calculates the MD5 digest of a string as a 32-character hexadecimal string.
* **sha1()**: Calculates the SHA-1 digest of a string as a 40-character hexadecimal string.
* **sha2()**: Calculates the SHA-2 family of hash functions (sha224, sha256, sha384, sha512).
* **crc32()**: Computes a cyclic redundancy check (CRC32) of a string.
* **xxhash64()**: Computes a 64-bit hash using the xxHash algorithm.

**Window Functions**

Functions that operate over a window of rows (often used in conjunction with Window specifications).

* row\_number(): Assigns a unique row number to each row within a window partition.
* rank(): Returns the rank of rows within a window partition.
* dense\_rank(): Returns the dense rank of rows within a window partition.
* ntile(): Divides rows into a specified number of roughly equal groups.
* lead(): Returns the value from the next row in the window.
* lag(): Returns the value from the previous row in the window.
* cume\_dist(): Returns the cumulative distribution of values within a window partition.
* percent\_rank(): Returns the relative rank of a row as a percentage.

**Null Handling**

Functions for handling null values.

* isnull(): Returns true if the column is null.
* isnan(): Returns true if the column contains NaN (Not a Number).
* coalesce(): Returns the first non-null value.
* na.fill(): Replaces null values with a specified value.
* na.drop(): Drops rows with null values.
* na.replace(): Replaces values in a column with other values.

**Miscellaneous Functions**

Other useful functions that don't fit neatly into the above categories.

* lit(): Creates a column of a literal value.
* col(): Returns a column based on a string name.
* when(): A conditional expression (similar to SQL CASE WHEN).
* expr(): Parses the expression string into a column.
* monotonically\_increasing\_id(): Returns a column that generates unique increasing 64-bit integers.
* input\_file\_name(): Returns the name of the file being read.
* struct(): Creates a struct column from multiple columns.