iDataValidator Functions

[validatequestion 2](#_Toc174498331)

[validatecolumns 4](#_Toc174498332)

[checkinvalid 5](#_Toc174498333)

[checkblanks 6](#_Toc174498334)

[checknoblanks 6](#_Toc174498335)

[checkinvalid\_logic 7](#_Toc174498336)

[checkvalid\_logic 7](#_Toc174498337)

[checkmasking 8](#_Toc174498338)

[checkquestion\_mask 9](#_Toc174498339)

[checkrank 10](#_Toc174498340)

[checksum 11](#_Toc174498341)

[checksum100 12](#_Toc174498342)

[checkterm 13](#_Toc174498343)

## validatequestion

Validate data based on question type, value range, exclusivity, completeness, text length requirements, and custom validation.

**Parameters:**

* question (Question): The question object with properties like type, datacols, etc.
* qtype (str): The expected question type (default: 'single').
* valid\_values (list, np.ndarray or Callable): List, array, or function returning valid values for 'single' and 'multiple' question types.
* exclusive (list): List of column names that are exclusive.
* optional\_cols (list): List of columns where blanks are allowed as valid entries.
* exclude\_cols (list): List of column names to exclude from validation.
* range\_param (RangeTuple or Callable): A tuple specifying lower and upper range or a function returning such a tuple.
* allow\_blanks (bool): Whether to allow blank values as valid.
* **required (int): If set to 1, at least one of the columns must have a non** - zero/non-null entry.
* at\_most (int): Maximum number of selections allowed.
* at\_least (int): Minimum number of selections required.
* txt\_min\_length (int): Minimum length of text after removing spaces and special characters, if applicable.
* txt\_max\_length (int): Maximum length of text after removing spaces and special characters, if applicable.
* custom\_row\_validation (function): A custom validation function that takes a row as an argument.
* condition (function): A function that takes a row and returns True /False
* skip\_check\_blank (bool): Whether to skip checking for blanks in rows that do not meet the condition

Returns: None

Usage examples:

*# Define a custom row validation function*

*def qcustom\_row\_validation\_function(row):*

*# Custom validation logic*

*for column in row.index:*

*if column != 'record' and row[column] % 2 != 0:*

*adderror(row['record'], column, row[column], 'Custom validation: Value is not even')*

*# Use a lambda function as a condition*

*condition = lambda row: row['S1'] in range(1, 5)*

*# Call with custom row validation function and condition*

*validatequestion(question, qtype='number', range\_param=(0, 100), custom\_row\_validation=custom\_row\_validation\_function, condition=condition)*

## validatecolumns

Validate data based on columns, value range, exclusivity, completeness, text length requirements, and custom validation.

**Parameters:**

* question\_id (str): The question id to be tagged in error log
* srcdatacols (list[str]): column names to check as list of strings
* columns\_type (str): The expected column type (default: 'single').
* valid\_values (list, np.ndarray or Callable): List, array, or function returning valid values for 'single' and 'multiple' question types.
* exclusive (list): List of column names that are exclusive.
* optional\_cols (list): List of columns where blanks are allowed as valid entries.
* exclude\_cols (list): List of column names to exclude from validation.
* range\_param (RangeTuple or Callable): A tuple specifying lower and upper range or a function returning such a tuple.
* allow\_blanks (bool): Whether to allow blank values as valid.
* required (int): If set to 1, at least one of the columns must have a non - zero/non-null entry.
* at\_most (int): Maximum number of selections allowed.
* at\_least (int): Minimum number of selections required.
* txt\_min\_length (int): Minimum length of text after removing spaces and special characters, if applicable.
* txt\_max\_length (int): Maximum length of text after removing spaces and special characters, if applicable.
* custom\_row\_validation (function): A custom validation function that takes a row as an argument.
* condition (function): A function that takes a row and returns True /False
* skip\_check\_blank (bool): Whether to skip checking for blanks in rows that do not meet the condition

## checkinvalid

Check for invalid values in set of columns - - if any column statisfies invalid condition will be added to error log

**Parameters:**

* datacols list(str) : list column names to be checked for invalid values
* columns\_type (str) : single, multi or text
* invalid\_values (list, np.ndarray or Callable): List, array, or function returning valid values for 'single' and 'multiple' question types.
* blank\_as\_invalid (bool): Whether to allow blank values as invalid.
* exclude\_cols (list): List of column names to exclude from validation.
* invalid\_range\_value: (RangeTuple or Callable): A tuple specifying lower and upper range or a function returning such a tuple.
* condition (function): A function that takes a row and returns True /False

**Usage:**

* checkInvalidLogic(label, condition)

## checkblanks

Check if specified columns contain non blank values.

**Parameters:**

* cols\_to\_check (str or list or question): Column name or list of column names to check for blank values.
* condition (function): A function that takes a row and returns True /False

**Usage:**

* checkBlanks('A'))
* checkBlanks(question)
* checkBlanks(['A','B']))

## checknoblanks

Check if specified columns contain blank (NaN) values.

**Parameters:**

* cols\_to\_check (str or list or question): Column name or list of column names to check for blank values.
* condition (function): A function that takes a row and returns True /False

**Usage:**

* checknoblanks('A'))
* checknoblanks(question)
* checknoblanks(['A','B']))

## checkinvalid\_logic

Check for invalid logic across all records - - if any records statisfied, will be add to error log

**Parameters:**

* logic\_label (str) : Label to log in error log
* condition (function): A function that takes a row and returns True /False

**Usage:**

* checkInvalidLogic(label, condition)

## checkvalid\_logic

Check for valid logic across all records - - if any records not statisfied, will be add to error log

**Parameters:**

* logic\_label (str) : Label to log in error log
* condition (function): A function that takes a row and returns True /False

**Usage:**

* checkValidLogic(label, condition)

## checkmasking

Check if specified columns contain values that meet a specific condition and log errors.

**Parameters:**

* question\_cols (list): List of columns to check.
* maskcond\_cols (list): List of columns to check conditions on.
* condition (str): Condition to check on columns ('=', 'in', '>').
* always\_showcols (list): List of columns that should always have any value including zero.
* condition (function): A function that takes a row and returns True /False

**Usage:**

* checkmask(['A', 'B'], ['ConditionCol1', 'ConditionCol2'], '=1', ['C', 'D'])
* checkmask(['A', 'B'], ['ConditionCol1', 'ConditionCol2'], 'in[1, 2]', ['C', 'D'])
* checkmask(['A', 'B'], ['ConditionCol1', 'ConditionCol2'], '>0', ['C', 'D'])

## checkquestion\_mask

Check if question contain values that meet a specific condition and log errors.

**Parameters:**

* question (Question): Question object
* mask\_question (Question): Question object
* condition (str): Condition to check on columns ('=', 'in', '>').
* always\_showcols (list): List of columns that should always have any value including zero.
* condition (function): A function that takes a row and returns True /False

Usage:

* checkquestion\_mask(question, mask\_question, '=1', ['C', 'D'])
* checkquestion\_mask(question, mask\_question, 'in[1, 2]', ['C', 'D'])
* checkquestion\_mask(question, mask\_question, '>0', ['C', 'D'])

## checkrank

Check for unique values across specified columns in a question, ensuring each value is present only once and falls within the specified dynamic rank range (min\_rank = 1).

**Parameters:**

* question (object): The question object containing datacols attribute (list of columns to check).
* max\_rank\_type (str): Type of max rank ('static', 'sum\_columns', 'column').
* max\_rank\_value (int, list, str): Max rank value depending on max\_rank\_type.
  + 'static': integer
  + 'sum\_columns': list of columns to sum for max rank
  + 'column': column name containing the max rank
* exclude\_cols (list): List of columns to exclude from checking.

Usage:

* checkrank(question, max\_rank\_type='static', max\_rank\_value=15)
* checkrank(question, max\_rank\_type='column', max\_rank\_value='MaxRank')
* checkrank(question, max\_rank\_type='sum\_columns', max\_rank\_value=['A', 'B'])

## checksum

Check for the sum of values in the quesiton is eqial to or in the range provided.

**Parameters:**

* question (object): The question object containing datacols attribute (list of columns to sum).
* range\_type (str): Type of range ('static', 'column', 'sum\_columns').
* range\_value (tuple, str, list): Range values depending on range\_type.
  + 'static': (min, max)
  + 'column': column name containing the range
  + 'sum\_columns': list of columns to sum for the range
* exclude\_cols (list): List of columns to exclude from summing.

**Usage:**

* checksum(question, range\_type='static', range\_value=(5, 15)))
* checksum(question, range\_type='column', range\_value='RangeColumn')
* checksum(question, range\_type='sum\_columns', range\_value=['A', 'C'])
* checksum(question, range\_type='static', range\_value=(5, 15), exclude\_cols=['C'])

## checksum100

Check for the sum of values in the quesiton is eqial to 100

**Parameters:**

* question (object): The question object containing datacols attribute (list of columns to sum).
* exclude\_cols (list): List of columns to exclude from summing.

**Usage:**

* checksum100(question)
* checksum100(question, exclude\_cols=['C'])

## checkterm

Check for termination logic - - no records should be in the data satisfying this condition. If there, add to error log.

**Parameters:**

* term\_label: A label to add in error log if termination fails
* condition : A function that takes a row and returns True if the validation should be applied

**Usage:**

* checkterm(term\_label, condition=custom\_row\_validation\_function )