python-duplicate

Release 0.1.0

Clément Omont--Agnes

Feb 02, 2020

CONTENTS

The last release is available on PyPI and can be installed with `pip`:

```
$ pip install python-duplicate
```

CHAPTER

ONE

EXAMPLES

1.1 Data

Import the class

```
from pyduplicate import FromList
```

Create unique items from a list of number

```
lst = [1.5, 2, 3, 3, 4, 5, 6, 3, 1.5, 4]

from_list = FromList(lst)
returned_lst = from_list.create_unique()

# It will return:
# [1.5, 2, 3, 4, 5, 6]
```

Get unique items from a list of tuple

```
lst = [(1.5, 2), (3, 4), (3, 4), (5, 6), (3, 1.5), (3, 4)]

from_list = FromList(lst)
returned_lst = from_list.get_unique()

# It will return:
# [(1.5, 2), (5, 6), (3, 1.5)]
```

Get duplicate items from a list of dict (on the key "id" only)

```
lst = [
    {"id": [1.5, 2], "value": 'value 0'}, {"id": [3, 4], "value": 'value 1'},
    {"id": [3, 4], "value": 'value 2'}, {"id": [5, 6], "value": 'value 3'},
    {"id": [3, 1.5], "value": 'value 5'}, {"id": [3, 4], "value": 'value 6'}
]

from_list = FromList(lst, key='id')
returned_lst = from_list.get_duplicate()

# It will return:
# [
#   {"id": [3, 4], "value": 'value 1'},
#   {"id": [3, 4], "value": 'value 2'},
#   {"id": [3, 4], "value": 'value 5'}
# ]
```

Get feedback of unique items of a list of string

```
lst = [
    "one comma five", "two", "three", "three", "four", "five", "six",
    "three", "one comma five", "four"
]

from_list = FromList(lst)
returned_lst = from_list.analyze('unique')  # or 'duplicate'

# It will return:
# { "all_index": [1, 5, 6], "two": [1], "five": [5], "six": [6] }
```

1.2 Database

All examples below will use this table

```
CREATE TABLE `address` (
    `id` SMALLINT PRIMARY KEY,
    `complete_address` VARCHAR (255) NOT NULL,
    `country_id` SMALLINT REFERENCES country(id),
    `resident_id` SMALLINT REFERENCES resident(id)
);
```

Import the class and define dictionary to connect on the Database It use the same naming convention as Django, so you can directly pass the Django conf dict (All example are done with MySQL, but it's exactly the same with PostgreSQL)

```
from pyduplicate import FromMySQL  # or FromPSQL for Postgres

INFO_DICT = {
    'HOST': "MYSQL_HOSTNAME",
    'DATABASE': "MYSQL_DATABASE",
    'USER': "MYSQL_USER",
    'PASSWORD': "MYSQL_PASSWORD",
    'PORT': int("MYSQL_PORT"),
}
```

Get row or primary key of duplicate column

```
# First param is the connection dict, second the table and third the column on which␣
↪to search
from_mysql = FromMySQL(INFO_DICT, 'address', 'complete_address')

result = self.from_mysql.select_duplicate(rows_list=True)
# It will return:
# (
#    (2, '3 street test', 0, 2),
#    (4, '5 street test', 1, 4),
#    (5, '5 street test', 0, 3),
#    (7, '3 street test', 0, 0)
# )

result = self.from_mysql.select_duplicate()
# It will return:
# [2, 4, 5, 7]
```

Get row or primary key of unique column

```
# First param is the connection dict, second the table and third the column on which␣
↪to search
from_mysql = FromMySQL(INFO_DICT, 'address', 'complete_address')

result = self.from_mysql.select_unique(rows_list=True)
# It will return:
# (
#    (0, '1 street test', 0, 0),
#    (1, '2 street test', 1, 1),
#    (3, '4 street test', 0, 3),
#    (6, '6 street test', 0, 4)
# )

result = self.from_mysql.select_unique()
# It will return:
# [0, 1, 3, 6]
```

CHAPTER

TWO

DATA DOCUMENTATION

This document specifies python-duplicate's data package.

2.1 FromList

class pyduplicate.data.from_list.**FromList**(*lst: list*, *key: str = None*)

Handle search of duplicate or unique item inside a list

Parameters

- **lst** (*list*) – Param to work on
- **key** (*str*) – Param to work in a list of dict if you need it

analyze(*analyze_type: str*, *feedback: bool = True*) → dict

Analyze the list to get the items and their indexes (depending on the analyze_type)

Parameters

- **analyze_type** (*str*) – Param to determine what type of analyse to do
- **feedback** (*bool*) – Param to return more or less information

Returns

- without feedback:

```
{ "all_index": [REVERSED_ORDER_INDEX(ES)] }
```

- with feedback:

```
{
    "all_index": [REVERSED_ORDER_INDEX(ES)],
    "VALUE_X": [INDEX(ES)],
    "VALUE_Y": [INDEX(ES)],
}
```

create_unique() → list

Returns Unique items (by deleting duplicate items)

get_duplicate() → list

Returns Only all duplicate items

get_unique() → list

Returns Only unique items

2.2 Utils

class pyduplicate.data.utils.**Utils**(*lst: list*)

Provides functions to handle search of duplicate or unique item inside a list

Parameters **lst** (*list*) – Param to work on. By default its the same list passed to *FromList*

create_list(*key: str*) → None

Create a list of all item for a given key

Parameters **key** (*str*) – Dict key on which to get data

create_unique_index(*feedback: bool = False*) → dict

Identify duplicated item and return the index of each of them except for the first one (in order to create a list of unique item)

Parameters **feedback** (*bool*) – Param to return more or less information

Returns

- without feedback:

```
{ "all_index": [REVERSED_ORDER_INDEX(ES)] }
```

- with feedback:

```
{
    "all_index": [REVERSED_ORDER_INDEX(ES)],
    "VALUE_X": [INDEX(ES)],
    "VALUE_Y": [INDEX(ES)],
}
```

create_update_feedback(*index: int*, *value: any*, *feedback: bool*) → None

Create or update the 'self.indexes' dict to provide feedback

get_indexes(*index_type: str*, *feedback: bool = False*) → dict

Identify unique or duplicate item and return the index of each of them

Parameters

- **index_type** (*str*) – Param to know what type of item index we want
- **feedback** (*bool*) – Param to return more or less information

Returns

- without feedback:

```
{ "all_index": [INDEX(ES)] }
```

- with feedback:

```
{
    "all_index": [INDEX(ES)],
    "VALUE_X": [INDEX(ES)],
    "VALUE_Y": [INDEX(ES)],
}
```

get_type() → None

Get the type of the first item which is not None

validate_items() → None
Check if all items are of the same type

Raises **TypeError** – If items are NOT of the same type

CHAPTER

THREE

DATABASE DOCUMENTATION

This document specifies python-duplicate's database package.

3.1 FromPSQL

class pyduplicate.database.from_psql.**FromPSQL**(*info_dict: dict*, *table: str*, *column: str*)

Handle search of duplicate or unique item inside a PostgreSQL table

Parameters

- **info_dict** (*dict*) – Param to connect to PostgreSQL
- **table** (*str*) – Param to select the right table
- **column** (*str*) – Param to search duplicate or unique on

connect() → object

Connect to PostgreSQL (with information given to the class)

Returns psycopg2 connection object

disconnect() → None

Disconnect of PostgreSQL

get_pk_name() → str

Returns Primary key name

select_duplicate(*rows_list: bool = False*) → list

Parameters **rows_list** – Boolean to return list of pk or rows

Returns Duplicate entries

select_unique(*rows_list: bool = False*) → list

Parameters **rows_list** – Boolean to return list of pk or rows

Returns Unique entries

3.2 FromMySQL

class pyduplicate.database.from_mysql.**FromMySQL**(*info_dict: dict*, *table: str*, *column: str*)
Handle search of duplicate or unique item inside a MySQL table

Parameters

- **info_dict** (*dict*) – Param to connect to MySQL
- **table** (*str*) – Param to select the right table
- **column** (*str*) – Param to search duplicate or unique on

connect() → object
Connect to MySQL (with information given to the class)

Returns PyMySQL connection object

disconnect() → None
Disconnect of MySQL

get_pk_name() → str

Returns Primary key name

select_duplicate(*rows_list: bool = False*) → list

Parameters **rows_list** – Boolean to return list of pk or rows

Returns Duplicate entries

select_unique(*rows_list: bool = False*) → list

Parameters **rows_list** – Boolean to return list of pk or rows

Returns Unique entries

3.3 Utils

3.3.1 Postgres

class pyduplicate.database.utils.**Postgres**(*table: str*, *column: str*)
Provides functions to query PostgreSQL

Parameters

- **table** (*str*) – Param to select the right table
- **column** (*str*) – Param to search duplicate or unique on

3.3.2 MySQL

class pyduplicate.database.utils.**MySQL**(*table: str*, *column: str*)
Provides functions to query MySQL

Parameters

- **table** (*str*) – Param to select the right table
- **column** (*str*) – Param to search duplicate or unique on

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX