

---

# **pandagg Documentation**

***Release 0.1***

**Léonard Binet**

**Apr 14, 2020**



<b>1</b>	<b>Principles</b>	<b>1</b>
1.1	Elasticsearch tree structures . . . . .	1
1.2	Interactive usage . . . . .	1
<b>2</b>	<b>User Guide</b>	<b>3</b>
2.1	Query . . . . .	3
2.1.1	Pandagg DSL . . . . .	3
2.1.2	Chaining . . . . .	4
2.1.3	Regular syntax . . . . .	5
2.2	Aggregation . . . . .	5
2.2.1	Aggregation declaration . . . . .	5
2.2.2	Aggregation response . . . . .	5
2.3	Mapping . . . . .	5
2.3.1	Mapping DSL . . . . .	6
2.3.2	Interactive mapping . . . . .	7
2.4	Cluster indices discovery . . . . .	8
<b>3</b>	<b>Advanced usage</b>	<b>9</b>
<b>4</b>	<b>IMDB dataset</b>	<b>11</b>
4.1	Query requirements . . . . .	11
4.2	Data source . . . . .	11
4.3	Index mapping . . . . .	12
4.3.1	Overview . . . . .	12
4.3.2	Which fields require nesting? . . . . .	12
4.3.3	Text or keyword fields? . . . . .	12
4.3.4	Final mapping . . . . .	12
4.4	Steps to start playing with your index . . . . .	13
4.4.1	Dump tables . . . . .	13
4.4.2	Clone pandagg and setup environment . . . . .	13
4.4.3	Serialize movie documents and insert them . . . . .	14
4.4.4	Explore pandagg notebooks . . . . .	14
<b>5</b>	<b>pandagg package</b>	<b>15</b>
5.1	Subpackages . . . . .	15
5.1.1	pandagg.interactive package . . . . .	15
5.1.1.1	Submodules . . . . .	15

5.1.1.2	Module contents . . . . .	17
5.1.2	pandagg.node package . . . . .	17
5.1.2.1	Subpackages . . . . .	17
5.1.2.2	Submodules . . . . .	38
5.1.2.3	Module contents . . . . .	38
5.1.3	pandagg.tree package . . . . .	38
5.1.3.1	Submodules . . . . .	38
5.1.3.2	Module contents . . . . .	43
5.2	Submodules . . . . .	43
5.2.1	pandagg.agg module . . . . .	43
5.2.2	pandagg.client module . . . . .	49
5.2.3	pandagg.exceptions module . . . . .	49
5.2.4	pandagg.mapping module . . . . .	50
5.2.5	pandagg.query module . . . . .	55
5.2.6	pandagg.utils module . . . . .	61
5.3	Module contents . . . . .	61
<b>6</b>	<b>Contributing to Pandagg</b>	<b>63</b>
6.1	Our Development Process . . . . .	63
6.2	Pull Requests . . . . .	63
6.3	Any contributions you make will be under the MIT Software License . . . . .	63
6.4	Issues . . . . .	64
6.5	Report bugs using Github's issues . . . . .	64
6.6	Write bug reports with detail, background, and sample code . . . . .	64
6.7	License . . . . .	64
6.8	References . . . . .	64
<b>7</b>	<b>Installing</b>	<b>65</b>
<b>8</b>	<b>Usage</b>	<b>67</b>
<b>9</b>	<b>License</b>	<b>69</b>
<b>10</b>	<b>Contributing</b>	<b>71</b>
	<b>Python Module Index</b>	<b>73</b>
	<b>Index</b>	<b>75</b>

---

**Note:** This is a work in progress. Some sections still need to be furnished.

---

**pandagg** is designed for both for “regular” code repository usage, and “interactive” usage (ipython or jupyter notebook usage with autocompletion features inspired by [pandas](#) design).

This library focuses on two principles:

- stick to the **tree** structure of Elasticsearch objects
- provide simple and flexible interfaces to make it easy and intuitive to use in an interactive usage

## 1.1 Elasticsearch tree structures

Many Elasticsearch objects have a **tree** structure, ie they are built from a hierarchy of **nodes**:

- a [mapping](#) (tree) is a hierarchy of [fields](#) (nodes)
- a [query](#) (tree) is a hierarchy of query clauses (nodes)
- an [aggregation](#) (tree) is a hierarchy of aggregation clauses (nodes)
- an aggregation response (tree) is a hierarchy of response buckets (nodes)

This library aims to stick to that structure by providing a flexible syntax distinguishing **trees** and **nodes**.

## 1.2 Interactive usage

Some classes are not intended to be used elsewhere than in interactive mode (ipython), since their purpose is to serve auto-completion features and convenient representations.

They won’t serve you for any other usage than interactive ones.

Namely:

- *pandagg.mapping.IMapping*: used to interactively navigate in mapping and run quick aggregations on some fields
- *pandagg.client.Elasticsearch*: used to discover cluster indices, and eventually navigate their mappings, or run quick access aggregations or queries.
- *pandagg.agg.AggregationResponse*: used to interactively navigate in an aggregation response

These use case will be detailed in following sections.

---

**Note:** Examples will be based on *IMDB dataset* data. This is a work in progress. Some sections still need to be furnished.

---

## 2.1 Query

The *Query* class allows multiple ways to declare and update an Elasticsearch query.

Let's explore the multiple ways we have to declare the following query:

```
>>> expected_query = {'bool': {'must': [  
>>>     {'terms': {'genres': ['Action', 'Thriller']}},  
>>>     {'range': {'rank': {'gte': 7}}},  
>>>     {'nested': {  
>>>         'path': 'roles',  
>>>         'query': {'bool': {'must': [  
>>>             {'term': {'roles.gender': {'value': 'F'}}},  
>>>             {'term': {'roles.role': {'value': 'Reporter'}}]}}  
>>>         }  
>>>     ]}}  
>>> ]}}
```

### 2.1.1 Pandagg DSL

Pandagg provides a DSL to declare this query in a quite similar fashion:

```
>>> from pandagg.query import Nested, Bool, Query, Range, Term, Terms
```

```
>>> q = Query(  
>>>     Bool(must=[  
>>>         Terms('genres', terms=['Action', 'Thriller']),  
>>>         Range('rank', gte=7),  
>>>         Nested(  
>>>             path='roles',  
>>>             query=Bool(must=[  
>>>                 Term('roles.gender', value='F'),  
>>>                 Term('roles.role', value='Reporter')  
>>>             ]) )  
>>>     ]) )  
>>> )
```

The serialized query is then available with `query_dict` method:

```
>>> q.query_dict() == expected_query  
True
```

A visual representation of the query helps to have a clearer view:

```
>>> q  
<Query>  
bool  
├─ must  
│   └─ nested  
│       └─ path="roles"  
│           └─ query  
│               └─ bool  
│                   └─ must  
│                       └─ term, field=roles.gender, value="F"  
│                       └─ term, field=roles.role, value="Reporter"  
└─ range, field=rank, gte=7  
└─ terms, field=genres, values=['Action', 'Thriller']
```

## 2.1.2 Chaining

Another way to declare this query is through chaining:

```
>>> from pandagg.utils import equal_queries  
>>> from pandagg.query import Nested, Bool, Query, Range, Term, Terms
```

```
>>> q = Query()\  
>>>     .query({'terms': {'genres': ['Action', 'Thriller']}})\  
>>>     .nested(path='roles', _name='nested_roles', query=Term('roles.gender', value=  
    ↪ 'F'))\  
>>>     .query(Range('rank', gte=7))\  
>>>     .query(Term('roles.role', value='Reporter'), parent='nested_roles')
```

```
>>> equal_queries(q.query_dict(), expected_query)  
True
```

---

**Note:** `equal_queries` function won't consider order of clauses in must/should parameters since it actually doesn't matter in Elasticsearch execution, ie



```
>>> equal_queries({'must': [A, B]}, {'must': [B, A]})
True
```

### 2.1.3 Regular syntax

Eventually, you can also use regular Elasticsearch dict syntax:

```
>>> q = Query(expected_query)
>>> q
<Query>
bool
├── must
│   ├── nested
│   │   ├── path="roles"
│   │   └── query
│   │       └── bool
│   │           └── must
│   │               ├── term, field=roles.gender, value="F"
│   │               └── term, field=roles.role, value="Reporter"
│   └── range, field=rank, gte=7
└── terms, field=genres, values=['Action', 'Thriller']
```

## 2.2 Aggregation

### 2.2.1 Aggregation declaration

### 2.2.2 Aggregation response

TODO

## 2.3 Mapping

Here is a portion of *IMDB dataset* example mapping:

```
>>> imdb_mapping = {
>>>     'dynamic': False,
>>>     'properties': {
>>>         'movie_id': {'type': 'integer'},
>>>         'name': {
>>>             'type': 'text',
>>>             'fields': {
>>>                 'raw': {'type': 'keyword'}
>>>             }
>>>         },
>>>         'year': {
>>>             'type': 'date',
>>>             'format': 'yyyy'
>>>         },
>>>         'rank': {'type': 'float'},
```

(continues on next page)

(continued from previous page)

```

>>>     'genres': {'type': 'keyword'},
>>>     'roles': {
>>>         'type': 'nested',
>>>         'properties': {
>>>             'role': {'type': 'keyword'},
>>>             'actor_id': {'type': 'integer'},
>>>             'gender': {'type': 'keyword'},
>>>             'first_name': {
>>>                 'type': 'text',
>>>                 'fields': {
>>>                     'raw': {'type': 'keyword'}
>>>                 }
>>>             },
>>>             'last_name': {
>>>                 'type': 'text',
>>>                 'fields': {
>>>                     'raw': {'type': 'keyword'}
>>>                 }
>>>             }
>>>         }
>>>     }
>>> }

```

### 2.3.1 Mapping DSL

The *Mapping* class provides a more compact view, which can help when dealing with large mappings:

```

>>> from pandagg.mapping import Mapping
>>> m = Mapping(imdb_mapping)
<Mapping>

```

— genres	{Object}
— movie_id	Keyword
— name	Integer
└ raw	Text
— rank	~ Keyword
— roles	Float
└ actor_id	[Nested]
└ first_name	Integer
└ raw	Text
└ gender	~ Keyword
└ last_name	Keyword
└ raw	Text
└ role	~ Keyword
└ year	Keyword
	Date

With pandagg DSL, an equivalent declaration would be the following:

```

>>> from pandagg.mapping import Mapping, Object, Nested, Float, Keyword, Date, \
↳ Integer, Text
>>>
>>> dsl_mapping = Mapping(properties=[
>>>     Integer('movie_id'),
>>>     Text('name', fields=[

```

(continues on next page)

(continued from previous page)

```

>>>     Keyword('raw')
>>>   ]),
>>>   Date('year', format='yyyy'),
>>>   Float('rank'),
>>>   Keyword('genres'),
>>>   Nested('roles', properties=[
>>>       Keyword('role'),
>>>       Integer('actor_id'),
>>>       Keyword('gender'),
>>>       Text('first_name', fields=[
>>>           Keyword('raw')
>>>       ]),
>>>       Text('last_name', fields=[
>>>           Keyword('raw')
>>>       ])
>>>   ])
>>> ])

```

Which is exactly equivalent to initial mapping:

```

>>> dsl_mapping.serialize() == imdb_mapping
True

```

### 2.3.2 Interactive mapping

In interactive context, the *IMapping* class provides navigation features with autocompletion to quickly discover a large mapping:

```

>>> from pandagg.mapping import IMapping
>>> m = IMapping(imdb_mapping)
>>> m.roles
<IMapping subpart: roles>
roles
├── actor_id           [Nested] Integer
├── first_name        Text
│   └── raw           ~ Keyword
├── gender            Keyword
├── last_name         Text
│   └── raw           ~ Keyword
└── role              Keyword
>>> m.roles.first_name
<IMapping subpart: roles.first_name>
first_name           Text
└── raw               ~ Keyword

```

To get the complete field definition, just call it:

```

>>> m.roles.first_name()
<Mapping Field first_name> of type text:
{
  "type": "text",
  "fields": {
    "raw": {
      "type": "keyword"
    }
  }
}

```

(continues on next page)

(continued from previous page)

```
}  
}
```

A **IMapping** instance can be bound to an Elasticsearch client to get quick access to aggregations computation on mapping fields.

Suppose you have the following client:

```
>>> from elasticsearch import Elasticsearch  
>>> client = Elasticsearch(hosts=['localhost:9200'])
```

Client can be bound either at initiation:

```
>>> m = IMapping(imdb_mapping, client=client, index_name='movies')
```

or afterwards through *bind* method:

```
>>> m = IMapping(imdb_mapping)  
>>> m.bind(client=client, index_name='movies')
```

Doing so will generate a **a** attribute on mapping fields, this attribute will list all available aggregation for that field type (with autocompletion):

```
>>> m.roles.gender.a.terms()  
[('M', {'key': 'M', 'doc_count': 2296792}),  
 ('F', {'key': 'F', 'doc_count': 1135174})]
```

---

**Note:** Nested clauses will be automatically taken into account.

---

## 2.4 Cluster indices discovery

TODO

## CHAPTER 3

---

### Advanced usage

---

---

**Note:** This is a work in progress. Some sections still need to be furnished.

- node and tree deserialization order
  - compound query insertion
-



You might know the Internet Movie Database, commonly called [IMDB](#).

Well it's a good simple example to showcase ElasticSearch capabilities.

In this case, relational databases (SQL) are a good fit to store with consistence this kind of data. Yet indexing some of this data in a optimized search engine will allow more powerful queries.

### 4.1 Query requirements

In this example, we'll suppose most usage/queries requirements will be around the concept of movie (rather than usages focused on fetching actors or directors, even though it will still be possible with this data structure).

The index should provide good performances trying to answer these kind question (non-exhaustive):

- in which movies this actor played?
- what movies genres were most popular among decades?
- which actors have played in best-rated movies, or worst-rated movies?
- which actors movies directors prefer to cast in their movies?
- which are best ranked movies of last decade in Action or Documentary genres?
- ...

### 4.2 Data source

I exported following SQL tables from MariaDB [following these instructions](#).

Relational schema is the following:

imdb tables

## 4.3 Index mapping

### 4.3.1 Overview

The base unit (document) will be a movie, having a name, rank (ratings), year of release, a list of actors and a list of directors.

Schematically:

```
Movie:
- name
- year
- rank
- [] genres
- [] directors
- [] actor roles
```

### 4.3.2 Which fields require nesting?

Since genres contain a single keyword field, in no case we need it to be stored as a nested field. On the contrary, actor roles and directors require a nested mapping if we consider applying multiple simultaneous query clauses on their sub-fields (for instance search movie in which actor is a woman AND whose role is nurse). More information on distinction between array and nested fields [here](#).

### 4.3.3 Text or keyword fields?

Some fields are easy to choose, in no situation gender will require a full text search, thus we'll store it as a keyword. On the other hand actors and directors names (first and last) will require full-text search, we'll thus opt for a text field. Yet we might want to aggregate on exact keywords to count number of movies per actor for instance. More information on distinction between text and keyword fields [here](#)

### 4.3.4 Final mapping

*TODO -> use `copy_to` parameter to build `full_name`*

.	
├ directors	[Nested]
│ └ director_id	Integer
│ └ first_name	Text
│ │ └ raw	~ Keyword
│ └ full_name	Text
│ │ └ raw	~ Keyword
│ └ genres	Keyword
│ └ last_name	Text
│ │ └ raw	~ Keyword
├ genres	Keyword
├ movie_id	Integer
├ name	Text
│ └ raw	~ Keyword
├ rank	Float
├ roles	[Nested]
│ └ actor_id	Integer

(continues on next page)



(continued from previous page)

├── first_name	Text
│   └── raw	~ Keyword
├── full_name	Text
│   └── raw	~ Keyword
├── gender	Keyword
├── last_name	Text
│   └── raw	~ Keyword
├── role	Keyword
└── year	Date

## 4.4 Steps to start playing with your index

Note to Elastic, if you have a spare cluster to prepare demo indices on which you could let your community perform read operations we could skip this step ;)

### 4.4.1 Dump tables

Follow instruction on bottom of <https://relational.fit.cvut.cz/dataset/IMDb> page and dump following tables in a directory:

- movies.csv
- movies\_genres.csv
- movies\_directors.csv
- directors.csv
- directors\_genres.csv
- roles.csv
- actors.csv

### 4.4.2 Clone pandagg and setup environment

```
git clone git@github.com:alkemics/pandagg.git
cd pandagg

virtualenv env
python setup.py develop
pip install pandas simplejson jupyter seaborn
```

Then copy `conf.py.dist` file into `conf.py` and edit variables as suits you, for instance:

```
# your cluster address
ES_HOST = 'localhost:9200'

# where your table dumps are stored, and where serialized output will be written
DATA_DIR = '/path/to/dumps/'
OUTPUT_FILE_NAME = 'serialized.json'
```

### 4.4.3 Serialize movie documents and insert them

```
# generate serialized movies documents, ready to be inserted in ES
# can take a while
python examples/imdb/serialize.py

# create index with mapping if necessary, bulk insert documents in ES
python examples/imdb/load.py
```

### 4.4.4 Explore pandagg notebooks

An example notebook is available to showcase some of pandagg functionalities: [here it is](#).

Code is present in `examples/imdb/IMDB_exploration.py` file.

## 5.1 Subpackages

### 5.1.1 pandagg.interactive package

#### 5.1.1.1 Submodules

##### pandagg.interactive.abstract module

```
class pandagg.interactive.abstract.Obj(**kwargs)
```

Bases: `object`

Object class that allows to get items both by attribute `__getattr__` access: `obj.attribute` or by dict `__getitem__` access: `>>> obj = Obj(key='value') >>> obj.key 'value' >>> obj['key'] 'value'`

In Ipython interpreter, attributes will be available in autocomplete (except private ones): `>>> obj = Obj(key='value', key2='value2') >>> obj.k` # press tab for autocompletion key key2

Items names that are not compliant with python attributes (accepted characters are [[a-zA-Z0-9\\_](#)] without beginning with a figure), will be only available through dict `__getitem__` access.

```
class pandagg.interactive.abstract.TreeBasedObj(tree, root_path=None, depth=1, initial_tree=None)
```

Bases: `pandagg.interactive.abstract.Obj`

Recursive Obj whose structure is defined by a `treelib.Tree` object.

The main purpose of this object is to iteratively expand the tree as attributes of this object. To avoid creating useless instances, only direct children of accessed nodes are expanded.

```
pandagg.interactive.abstract.is_valid_attr_name(item)
```

### pandagg.interactive.client module

```
class pandagg.interactive.client.Elasticsearch (hosts=None, transport_class=<class  
                                         'elasticsearch.transport.Transport'>,  
                                         **kwargs)
```

Bases: `elasticsearch.client.Elasticsearch`

```
fetch_indices (index='*')
```

**Parameters** **index** – Comma-separated list or wildcard expression of index names used to limit the request.

### pandagg.interactive.index module

```
class pandagg.interactive.index.Aliases (**kwargs)
```

Bases: `pandagg.interactive.abstract.Obj`

```
class pandagg.interactive.index.Index (name, settings, mapping, aliases, client=None)
```

Bases: `pandagg.interactive.abstract.Obj`

```
agg (arg, insert_below=None, output='dataframe', execute=True, **kwargs)
```

```
groupby (by, insert_below=None, **kwargs)
```

```
query (query, validate=False, **kwargs)
```

```
set_mapping (mapping)
```

```
class pandagg.interactive.index.Indices (**kwargs)
```

Bases: `pandagg.interactive.abstract.Obj`

### pandagg.interactive.mapping module

```
class pandagg.interactive.mapping.IMapping (from_=None, properties=None, dy-  
                                         namic=False, client=None, root_path=None,  
                                         depth=1, initial_tree=None, in-  
                                         dex_name=None)
```

Bases: `pandagg.interactive.abstract.TreeBasedObj`

Interactive wrapper upon mapping tree.

```
pandagg.interactive.mapping.as_mapping (mapping)
```

### pandagg.interactive.response module

```
class pandagg.interactive.response.IResponse (tree, client=None, index_name=None,  
                                         root_path=None, depth=None, ini-  
                                         tial_tree=None, query=None)
```

Bases: `pandagg.interactive.abstract.TreeBasedObj`

Interactive aggregation response.

```
get_bucket_filter ()
```

Build filters to select documents belonging to that bucket

```
list_documents (size=None, execute=True, _source=None, compact=False, **kwargs)
```

Return ES aggregation query to list documents belonging to given bucket. :param size: number of returned documents (ES default: 20) :param execute: if set to False, return aggregation query :param \_source: list of

desired documents attributes :param compact: provide more compact ES response :param kwargs: query arguments passed to aggregation body :return:

### 5.1.1.2 Module contents

## 5.1.2 pandagg.node package

### 5.1.2.1 Subpackages

#### pandagg.node.agg package

#### Submodules

#### pandagg.node.agg.abstract module

```
class pandagg.node.agg.abstract.AggNode (name, meta=None, **body)
    Bases: pandagg.node._node.Node

    Wrapper around elasticsearch aggregation concept. https://www.elastic.co/guide/en/elasticsearch/reference/2.3/search-aggregations.html

    Each aggregation can be seen both a Node that can be encapsulated in a parent agg.

    Define a method to build aggregation request.

    BLACKLISTED_MAPPING_TYPES = None

    KEY = NotImplementedError()

    VALUE_ATTRS = NotImplementedError()

    WHITELISTED_MAPPING_TYPES = None

    classmethod deserialize (name, body, meta=None)

    classmethod extract_bucket_value (response, value_as_dict=False)

    extract_buckets (response_value)

    get_filter (key)
        Return filter query to list documents having this aggregation key. :param key: string :return: elasticsearch filter query

    query_dict (with_name=False)
        ElasticSearch aggregation queries follow this formatting: {
            "<aggregation_name>" [{
                "<aggregation_type>" [{ <aggregation_body>
                } ], "meta": { [<meta_data_body> ] } ]?
            }
        }

        Query dict returns the following part (without aggregation name): {
            "<aggregation_type>" [{ <aggregation_body>
            } ], "meta": { [<meta_data_body> ] } ]?
```

```
    }
    tag
        The readable node name for human. This attribute can be accessed and modified with . and = operator respectively.

    classmethod valid_on_field_type (field_type)
class pandagg.node.agg.abstract.BucketAggNode (name, meta=None, aggs=None, **body)
    Bases: pandagg.node.agg.abstract.AgNode

    Bucket aggregation have special abilities: they can encapsulate other aggregations as children. Each time, the extracted value is a 'doc_count'.

    Provide methods: - to build aggregation request (with children aggregations) - to to extract buckets from raw response - to build query to filter documents belonging to that bucket

    Note: the aggs attribute's only purpose is for children initiation with the following syntax: >>> from pandagg.agg import Terms, Avg >>> agg = Terms( >>> name='term_agg', >>> field='some_path', >>> aggs=[ >>> Avg(agg_name='avg_agg', field='some_other_path') >>> ] >>> ) Yet, the aggs attribute will then be reset to None to avoid confusion since the real hierarchy is stored in the bpointer/fpointer attributes inherited from treelib.Tree class.

    VALUE_ATTRS = NotImplementedError()

    extract_buckets (response_value)

    get_filter (key)
        Provide filter to get documents belonging to document of given key.

class pandagg.node.agg.abstract.FieldOrScriptMetricAgg (name, meta=None, **body)
    Bases: pandagg.node.agg.abstract.MetricAgg

    Metric aggregation based on single field.

    VALUE_ATTRS = NotImplementedError()

class pandagg.node.agg.abstract.MetricAgg (name, meta=None, **body)
    Bases: pandagg.node.agg.abstract.AgNode

    Metric aggregation are aggregations providing a single bucket, with value attributes to be extracted.

    VALUE_ATTRS = NotImplementedError()

    extract_buckets (response_value)

    get_filter (key)
        Return filter query to list documents having this aggregation key. :param key: string :return: elasticsearch filter query

class pandagg.node.agg.abstract.MultipleBucketAgg (name,
                                                    key_path='key',
                                                    keyed=None,
                                                    meta=None,
                                                    aggs=None, **body)

    Bases: pandagg.node.agg.abstract.BucketAggNode

    IMPLICIT_KEYED = False

    VALUE_ATTRS = NotImplementedError()

    extract_buckets (response_value)

    get_filter (key)
        Provide filter to get documents belonging to document of given key.
```

```

class pandagg.node.agg.abstract.Pipeline(name, buckets_path, gap_policy=None,
                                          meta=None, aggs=None, **body)
    Bases: pandagg.node.agg.abstract.UniqueBucketAgg
    VALUE_ATTRS = NotImplementedError()
    get_filter(key)
        Provide filter to get documents belonging to document of given key.

class pandagg.node.agg.abstract.ScriptPipeline(name, script, buckets_path,
                                                gap_policy=None, meta=None,
                                                aggs=None, **body)
    Bases: pandagg.node.agg.abstract.Pipeline
    KEY = NotImplementedError()
    VALUE_ATTRS = 'value'

class pandagg.node.agg.abstract.ShadowRoot
    Bases: pandagg.node.agg.abstract.BucketAggNode
    Not a real aggregation.
    KEY = 'shadow_root'
    tag
        The readable node name for human. This attribute can be accessed and modified with . and = operator
        respectively.

class pandagg.node.agg.abstract.UniqueBucketAgg(name, meta=None, aggs=None,
                                                  **body)
    Bases: pandagg.node.agg.abstract.BucketAggNode
    Aggregations providing a single bucket.
    VALUE_ATTRS = NotImplementedError()
    extract_buckets(response_value)
    get_filter(key)
        Provide filter to get documents belonging to document of given key.

```

## pandagg.node.agg.bucket module

Not implemented aggregations include: - children agg - geo-distance - geo-hash grid - ipv4 - sampler - significant terms

```

class pandagg.node.agg.bucket.DateHistogram(name, field, interval=None, calendar_interval=None,
                                             fixed_interval=None, meta=None, keyed=False,
                                             key_as_string=True, aggs=None, **body)
    Bases: pandagg.node.agg.abstract.MultipleBucketAgg
    KEY = 'date_histogram'
    VALUE_ATTRS = ['doc_count']
    WHITELISTED_MAPPING_TYPES = ['date']
    get_filter(key)
        Provide filter to get documents belonging to document of given key.

```

```
class pandagg.node.agg.bucket.DateRange(name, field, key_as_string=True, aggs=None,
                                         meta=None, **body)
```

Bases: [pandagg.node.agg.bucket.Range](#)

```
KEY = 'date_range'
```

```
KEY_SEP = '::'
```

```
VALUE_ATTRS = ['doc_count']
```

```
WHITELISTED_MAPPING_TYPES = ['date']
```

```
class pandagg.node.agg.bucket.Filter(name, filter, meta=None, aggs=None, **body)
```

Bases: [pandagg.node.agg.abstract.UniqueBucketAgg](#)

```
KEY = 'filter'
```

```
VALUE_ATTRS = ['doc_count']
```

```
get_filter(key)
```

Provide filter to get documents belonging to document of given key.

```
class pandagg.node.agg.bucket.Filters(name, filters, other_bucket=False,
                                       other_bucket_key=None, meta=None, aggs=None,
                                       **body)
```

Bases: [pandagg.node.agg.abstract.MultipleBucketAgg](#)

```
DEFAULT_OTHER_KEY = '_other_'
```

```
IMPLICIT_KEYED = True
```

```
KEY = 'filters'
```

```
VALUE_ATTRS = ['doc_count']
```

```
get_filter(key)
```

Provide filter to get documents belonging to document of given key.

```
class pandagg.node.agg.bucket.Global(name, meta=None, aggs=None)
```

Bases: [pandagg.node.agg.abstract.UniqueBucketAgg](#)

```
KEY = 'global'
```

```
VALUE_ATTRS = ['doc_count']
```

```
get_filter(key)
```

Provide filter to get documents belonging to document of given key.

```
class pandagg.node.agg.bucket.Histogram(name, field, interval, meta=None, aggs=None,
                                         **body)
```

Bases: [pandagg.node.agg.abstract.MultipleBucketAgg](#)

```
KEY = 'histogram'
```

```
VALUE_ATTRS = ['doc_count']
```

```
WHITELISTED_MAPPING_TYPES = ['long', 'integer', 'short', 'byte', 'double', 'float', 'h
```

```
get_filter(key)
```

Provide filter to get documents belonging to document of given key.

```
class pandagg.node.agg.bucket.MatchAll(name, meta=None, aggs=None)
```

Bases: [pandagg.node.agg.bucket.Filter](#)

```
class pandagg.node.agg.bucket.Missing(name, field, meta=None, aggs=None, **body)
```

Bases: [pandagg.node.agg.abstract.UniqueBucketAgg](#)



```

BLACKLISTED_MAPPING_TYPES = []

KEY = 'missing'

VALUE_ATTRS = ['doc_count']

get_filter(key)
    Provide filter to get documents belonging to document of given key.

class pandagg.node.agg.bucket.Nested(name, path, meta=None, aggs=None)
    Bases: pandagg.node.agg.abstract.UniqueBucketAgg

    KEY = 'nested'

    VALUE_ATTRS = ['doc_count']

    WHITELISTED_MAPPING_TYPES = ['nested']

    get_filter(key)
        Provide filter to get documents belonging to document of given key.

class pandagg.node.agg.bucket.Range(name, field, ranges, keyed=False, meta=None,
                                     aggs=None, **body)
    Bases: pandagg.node.agg.abstract.MultipleBucketAgg

    KEY = 'range'

    KEY_SEP = '-'

    VALUE_ATTRS = ['doc_count']

    WHITELISTED_MAPPING_TYPES = ['long', 'integer', 'short', 'byte', 'double', 'float', 'h

    from_key

    get_filter(key)
        Provide filter to get documents belonging to document of given key.

    to_key

class pandagg.node.agg.bucket.ReverseNested(name, path=None, meta=None, aggs=None,
                                             **body)
    Bases: pandagg.node.agg.abstract.UniqueBucketAgg

    KEY = 'reverse_nested'

    VALUE_ATTRS = ['doc_count']

    WHITELISTED_MAPPING_TYPES = ['nested']

    get_filter(key)
        Provide filter to get documents belonging to document of given key.

class pandagg.node.agg.bucket.Terms(name, field, missing=None, size=None, aggs=None,
                                     meta=None, **body)
    Bases: pandagg.node.agg.abstract.MultipleBucketAgg

    Terms aggregation.

    BLACKLISTED_MAPPING_TYPES = []

    KEY = 'terms'

    VALUE_ATTRS = ['doc_count', 'doc_count_error_upper_bound', 'sum_other_doc_count']

    get_filter(key)
        Provide filter to get documents belonging to document of given key.

```

## pandagg.node.agg.deserializer module

pandagg.node.agg.deserializer.deserialize\_agg(*d*)

## pandagg.node.agg.metric module

```
class pandagg.node.agg.metric.Avg(name, meta=None, **body)
    Bases: pandagg.node.agg.abstract.FieldOrScriptMetricAgg
    KEY = 'avg'
    VALUE_ATTRS = ['value']
    WHITELISTED_MAPPING_TYPES = ['long', 'integer', 'short', 'byte', 'double', 'float', 'h

class pandagg.node.agg.metric.Cardinality(name, meta=None, **body)
    Bases: pandagg.node.agg.abstract.FieldOrScriptMetricAgg
    KEY = 'cardinality'
    VALUE_ATTRS = ['value']

class pandagg.node.agg.metric.ExtendedStats(name, meta=None, **body)
    Bases: pandagg.node.agg.abstract.FieldOrScriptMetricAgg
    KEY = 'extended_stats'
    VALUE_ATTRS = ['count', 'min', 'max', 'avg', 'sum', 'sum_of_squares', 'variance', 'std

    WHITELISTED_MAPPING_TYPES = ['long', 'integer', 'short', 'byte', 'double', 'float', 'h

class pandagg.node.agg.metric.GeoBound(name, meta=None, **body)
    Bases: pandagg.node.agg.abstract.FieldOrScriptMetricAgg
    KEY = 'geo_bounds'
    VALUE_ATTRS = ['bounds']
    WHITELISTED_MAPPING_TYPES = ['geo_point']

class pandagg.node.agg.metric.GeoCentroid(name, meta=None, **body)
    Bases: pandagg.node.agg.abstract.FieldOrScriptMetricAgg
    KEY = 'geo_centroid'
    VALUE_ATTRS = ['location']
    WHITELISTED_MAPPING_TYPES = ['geo_point']

class pandagg.node.agg.metric.Max(name, meta=None, **body)
    Bases: pandagg.node.agg.abstract.FieldOrScriptMetricAgg
    KEY = 'max'
    VALUE_ATTRS = ['value']
    WHITELISTED_MAPPING_TYPES = ['long', 'integer', 'short', 'byte', 'double', 'float', 'h

class pandagg.node.agg.metric.Min(name, meta=None, **body)
    Bases: pandagg.node.agg.abstract.FieldOrScriptMetricAgg
    KEY = 'min'
    VALUE_ATTRS = ['value']
    WHITELISTED_MAPPING_TYPES = ['long', 'integer', 'short', 'byte', 'double', 'float', 'h
```

```

class pandagg.node.agg.metric.PercentileRanks(name, field, values, meta=None, **body)
    Bases: pandagg.node.agg.abstract.FieldOrScriptMetricAgg

    KEY = 'percentile_ranks'

    VALUE_ATTRS = ['values']

    WHITELISTED_MAPPING_TYPES = ['long', 'integer', 'short', 'byte', 'double', 'float', 'h

class pandagg.node.agg.metric.Percentiles(name, meta=None, **body)
    Bases: pandagg.node.agg.abstract.FieldOrScriptMetricAgg

    Percents body argument can be passed to specify which percentiles to fetch.

    KEY = 'percentiles'

    VALUE_ATTRS = ['values']

    WHITELISTED_MAPPING_TYPES = ['long', 'integer', 'short', 'byte', 'double', 'float', 'h

class pandagg.node.agg.metric.Stats(name, meta=None, **body)
    Bases: pandagg.node.agg.abstract.FieldOrScriptMetricAgg

    KEY = 'stats'

    VALUE_ATTRS = ['count', 'min', 'max', 'avg', 'sum']

    WHITELISTED_MAPPING_TYPES = ['long', 'integer', 'short', 'byte', 'double', 'float', 'h

class pandagg.node.agg.metric.Sum(name, meta=None, **body)
    Bases: pandagg.node.agg.abstract.FieldOrScriptMetricAgg

    KEY = 'sum'

    VALUE_ATTRS = ['value']

    WHITELISTED_MAPPING_TYPES = ['long', 'integer', 'short', 'byte', 'double', 'float', 'h

class pandagg.node.agg.metric.TopHits(name, meta=None, **body)
    Bases: pandagg.node.agg.abstract.MetricAgg

    KEY = 'top_hits'

    VALUE_ATTRS = ['hits']

class pandagg.node.agg.metric.ValueCount(name, meta=None, **body)
    Bases: pandagg.node.agg.abstract.FieldOrScriptMetricAgg

    BLACKLISTED_MAPPING_TYPES = []

    KEY = 'value_count'

    VALUE_ATTRS = ['value']

```

## pandagg.node.agg.pipeline module

Pipeline aggregations: <https://www.elastic.co/guide/en/elasticsearch/reference/2.3/search-aggregations-pipeline.html>

```

class pandagg.node.agg.pipeline.AvgBucket(name, buckets_path, gap_policy=None,
                                          meta=None, aggs=None, **body)
    Bases: pandagg.node.agg.abstract.Pipeline

    KEY = 'avg_bucket'

    VALUE_ATTRS = ['value']

```

```
class pandagg.node.agg.pipeline.BucketScript (name, script, buckets_path,
                                              gap_policy=None, meta=None,
                                              aggs=None, **body)
    Bases: pandagg.node.agg.abstract.ScriptPipeline
    KEY = 'bucket_script'
    VALUE_ATTRS = ['value']

class pandagg.node.agg.pipeline.BucketSelector (name, script, buckets_path,
                                                gap_policy=None, meta=None,
                                                aggs=None, **body)
    Bases: pandagg.node.agg.abstract.ScriptPipeline
    KEY = 'bucket_selector'
    VALUE_ATTRS = None

class pandagg.node.agg.pipeline.BucketSort (name, script, buckets_path, gap_policy=None,
                                             meta=None, aggs=None, **body)
    Bases: pandagg.node.agg.abstract.ScriptPipeline
    KEY = 'bucket_sort'
    VALUE_ATTRS = None

class pandagg.node.agg.pipeline.CumulativeSum (name, buckets_path, gap_policy=None,
                                              meta=None, aggs=None, **body)
    Bases: pandagg.node.agg.abstract.Pipeline
    KEY = 'cumulative_sum'
    VALUE_ATTRS = ['value']

class pandagg.node.agg.pipeline.Derivative (name, buckets_path, gap_policy=None,
                                           meta=None, aggs=None, **body)
    Bases: pandagg.node.agg.abstract.Pipeline
    KEY = 'derivative'
    VALUE_ATTRS = ['value']

class pandagg.node.agg.pipeline.ExtendedStatsBucket (name, buckets_path,
                                                     gap_policy=None, meta=None,
                                                     aggs=None, **body)
    Bases: pandagg.node.agg.abstract.Pipeline
    KEY = 'extended_stats_bucket'
    VALUE_ATTRS = ['count', 'min', 'max', 'avg', 'sum', 'sum_of_squares', 'variance', 'std']

class pandagg.node.agg.pipeline.MaxBucket (name, buckets_path, gap_policy=None,
                                           meta=None, aggs=None, **body)
    Bases: pandagg.node.agg.abstract.Pipeline
    KEY = 'max_bucket'
    VALUE_ATTRS = ['value']

class pandagg.node.agg.pipeline.MinBucket (name, buckets_path, gap_policy=None,
                                           meta=None, aggs=None, **body)
    Bases: pandagg.node.agg.abstract.Pipeline
    KEY = 'min_bucket'
    VALUE_ATTRS = ['value']
```

```

class pandagg.node.agg.pipeline.MovingAvg(name, buckets_path, gap_policy=None,
                                           meta=None, aggs=None, **body)
    Bases: pandagg.node.agg.abstract.Pipeline
    KEY = 'moving_avg'
    VALUE_ATTRS = ['value']

class pandagg.node.agg.pipeline.PercentilesBucket(name, buckets_path,
                                                    gap_policy=None, meta=None,
                                                    aggs=None, **body)
    Bases: pandagg.node.agg.abstract.Pipeline
    KEY = 'percentiles_bucket'
    VALUE_ATTRS = ['values']

class pandagg.node.agg.pipeline.SerialDiff(name, buckets_path, gap_policy=None,
                                             meta=None, aggs=None, **body)
    Bases: pandagg.node.agg.abstract.Pipeline
    KEY = 'serial_diff'
    VALUE_ATTRS = ['value']

class pandagg.node.agg.pipeline.StatsBucket(name, buckets_path, gap_policy=None,
                                             meta=None, aggs=None, **body)
    Bases: pandagg.node.agg.abstract.Pipeline
    KEY = 'stats_bucket'
    VALUE_ATTRS = ['count', 'min', 'max', 'avg', 'sum']

class pandagg.node.agg.pipeline.SumBucket(name, buckets_path, gap_policy=None,
                                           meta=None, aggs=None, **body)
    Bases: pandagg.node.agg.abstract.Pipeline
    KEY = 'sum_bucket'
    VALUE_ATTRS = ['value']

```

## Module contents

### pandagg.node.mapping package

#### Submodules

#### pandagg.node.mapping.abstract module

```

class pandagg.node.mapping.abstract.Field(name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node._node.Node
    DISPLAY_PATTERN = '%s'
    KEY = NotImplementedError()
    body(with_children=False)
    classmethod deserialize(name, body, depth=0, is_subfield=False)
    reset_data()
    tree_repr

```

## pandagg.node.mapping.deserializer module

`pandagg.node.mapping.deserializer.deserialize_field(name, body, depth=0, is_subfield=False)`

## pandagg.node.mapping.field\_datatypes module

<https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping-types.html>

**class** `pandagg.node.mapping.field_datatypes.Alias` (*name, depth=0, is\_subfield=False, \*\*body*)

Bases: `pandagg.node.mapping.abstract.Field`

Defines an alias to an existing field.

**KEY** = 'alias'

**class** `pandagg.node.mapping.field_datatypes.Binary` (*name, depth=0, is\_subfield=False, \*\*body*)

Bases: `pandagg.node.mapping.abstract.Field`

**KEY** = 'binary'

**class** `pandagg.node.mapping.field_datatypes.Boolean` (*name, depth=0, is\_subfield=False, \*\*body*)

Bases: `pandagg.node.mapping.abstract.Field`

**KEY** = 'boolean'

**class** `pandagg.node.mapping.field_datatypes.Byte` (*name, depth=0, is\_subfield=False, \*\*body*)

Bases: `pandagg.node.mapping.abstract.Field`

**KEY** = 'byte'

**class** `pandagg.node.mapping.field_datatypes.Completion` (*name, depth=0, is\_subfield=False, \*\*body*)

Bases: `pandagg.node.mapping.abstract.Field`

To provide auto-complete suggestions

**KEY** = 'completion'

**class** `pandagg.node.mapping.field_datatypes.Date` (*name, depth=0, is\_subfield=False, \*\*body*)

Bases: `pandagg.node.mapping.abstract.Field`

**KEY** = 'date'

**class** `pandagg.node.mapping.field_datatypes.DateNanos` (*name, depth=0, is\_subfield=False, \*\*body*)

Bases: `pandagg.node.mapping.abstract.Field`

**KEY** = 'date\_nanos'

**class** `pandagg.node.mapping.field_datatypes.DateRange` (*name, depth=0, is\_subfield=False, \*\*body*)

Bases: `pandagg.node.mapping.abstract.Field`

**KEY** = 'date\_range'

**class** `pandagg.node.mapping.field_datatypes.DenseVector` (*name, depth=0, is\_subfield=False, \*\*body*)

Bases: `pandagg.node.mapping.abstract.Field`

Record dense vectors of float values.

**KEY = 'dense\_vector'**

```
class pandagg.node.mapping.field_datatypes.Double(name, depth=0, is_subfield=False,
                                                **body)
```

Bases: *pandagg.node.mapping.abstract.Field*

**KEY = 'double'**

```
class pandagg.node.mapping.field_datatypes.DoubleRange(name, depth=0,
                                                         is_subfield=False, **body)
```

Bases: *pandagg.node.mapping.abstract.Field*

**KEY = 'double\_range'**

```
class pandagg.node.mapping.field_datatypes.Flattened(name, depth=0,
                                                       is_subfield=False, **body)
```

Bases: *pandagg.node.mapping.abstract.Field*

Allows an entire JSON object to be indexed as a single field.

**KEY = 'flattened'**

```
class pandagg.node.mapping.field_datatypes.Float(name, depth=0, is_subfield=False,
                                                  **body)
```

Bases: *pandagg.node.mapping.abstract.Field*

**KEY = 'float'**

```
class pandagg.node.mapping.field_datatypes.FloatRange(name, depth=0,
                                                         is_subfield=False, **body)
```

Bases: *pandagg.node.mapping.abstract.Field*

**KEY = 'float\_range'**

```
class pandagg.node.mapping.field_datatypes.GeoPoint(name, depth=0,
                                                       is_subfield=False, **body)
```

Bases: *pandagg.node.mapping.abstract.Field*

For lat/lon points

**KEY = 'geo\_point'**

```
class pandagg.node.mapping.field_datatypes.GeoShape(name, depth=0,
                                                       is_subfield=False, **body)
```

Bases: *pandagg.node.mapping.abstract.Field*

For complex shapes like polygons

**KEY = 'geo\_shape'**

```
class pandagg.node.mapping.field_datatypes.HalfFloat(name, depth=0,
                                                         is_subfield=False, **body)
```

Bases: *pandagg.node.mapping.abstract.Field*

**KEY = 'half\_float'**

```
class pandagg.node.mapping.field_datatypes.Histogram(name, depth=0,
                                                         is_subfield=False, **body)
```

Bases: *pandagg.node.mapping.abstract.Field*

For pre-aggregated numerical values for percentiles aggregations.

**KEY = 'histogram'**

```
class pandagg.node.mapping.field_datatypes.IP (name,    depth=0,    is_subfield=False,
                                              **body)
    Bases: pandagg.node.mapping.abstract.Field
    for IPv4 and IPv6 addresses
    KEY = 'IP'

class pandagg.node.mapping.field_datatypes.Integer (name, depth=0, is_subfield=False,
                                              **body)
    Bases: pandagg.node.mapping.abstract.Field
    KEY = 'integer'

class pandagg.node.mapping.field_datatypes.IntegerRange (name,          depth=0,
                                                         is_subfield=False,
                                                         **body)
    Bases: pandagg.node.mapping.abstract.Field
    KEY = 'integer_range'

class pandagg.node.mapping.field_datatypes.Join (name,    depth=0,    is_subfield=False,
                                              **body)
    Bases: pandagg.node.mapping.abstract.Field
    Defines parent/child relation for documents within the same index
    KEY = 'join'

class pandagg.node.mapping.field_datatypes.Keyword (name, depth=0, is_subfield=False,
                                              **body)
    Bases: pandagg.node.mapping.abstract.Field
    KEY = 'keyword'

class pandagg.node.mapping.field_datatypes.Long (name,    depth=0,    is_subfield=False,
                                              **body)
    Bases: pandagg.node.mapping.abstract.Field
    KEY = 'long'

class pandagg.node.mapping.field_datatypes.LongRange (name,          depth=0,
                                                         is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field
    KEY = 'long_range'

class pandagg.node.mapping.field_datatypes.MapperAnnotatedText (name, depth=0,
                                                         is_subfield=False,
                                                         **body)
    Bases: pandagg.node.mapping.abstract.Field
    To index text containing special markup (typically used for identifying named entities)
    KEY = 'annotated-text'

class pandagg.node.mapping.field_datatypes.MapperMurMur3 (name,          depth=0,
                                                         is_subfield=False,
                                                         **body)
    Bases: pandagg.node.mapping.abstract.Field
    To compute hashes of values at index-time and store them in the index
    KEY = 'murmur3'
```



```

class pandagg.node.mapping.field_datatypes.Nested(name, depth=0, is_subfield=False,
                                                    **body)
    Bases: pandagg.node.mapping.abstract.Field
    DISPLAY_PATTERN = ' [%s]'
    KEY = 'nested'

class pandagg.node.mapping.field_datatypes.Object(name, depth=0, is_subfield=False,
                                                    **body)
    Bases: pandagg.node.mapping.abstract.Field
    DISPLAY_PATTERN = ' {%s}'
    KEY = 'object'

class pandagg.node.mapping.field_datatypes.Percolator(name, depth=0,
                                                        is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field
    Accepts queries from the query-dsl
    KEY = 'percolator'

class pandagg.node.mapping.field_datatypes.RankFeature(name, depth=0,
                                                        is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field
    Record numeric feature to boost hits at query time.
    KEY = 'rank_feature'

class pandagg.node.mapping.field_datatypes.RankFeatures(name, depth=0,
                                                         is_subfield=False,
                                                         **body)
    Bases: pandagg.node.mapping.abstract.Field
    Record numeric features to boost hits at query time.
    KEY = 'rank_features'

class pandagg.node.mapping.field_datatypes.ScaledFloat(name, depth=0,
                                                         is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field
    KEY = 'scaled_float'

class pandagg.node.mapping.field_datatypes.SearchAsYouType(name, depth=0,
                                                            is_subfield=False,
                                                            **body)
    Bases: pandagg.node.mapping.abstract.Field
    A text-like field optimized for queries to implement as-you-type completion
    KEY = 'search_as_you_type'

class pandagg.node.mapping.field_datatypes.Shape(name, depth=0, is_subfield=False,
                                                  **body)
    Bases: pandagg.node.mapping.abstract.Field
    For arbitrary cartesian geometries.
    KEY = 'shape'

class pandagg.node.mapping.field_datatypes.Short(name, depth=0, is_subfield=False,
                                                  **body)
    Bases: pandagg.node.mapping.abstract.Field

```

```
KEY = 'short'
```

**class** pandagg.node.mapping.field\_datatypes.**SparseVector** (*name*, *depth=0*,  
*is\_subfield=False*,  
*\*\*body*)

Bases: *pandagg.node.mapping.abstract.Field*

Record sparse vectors of float values.

```
KEY = 'sparse_vector'
```

**class** pandagg.node.mapping.field\_datatypes.**Text** (*name*, *depth=0*, *is\_subfield=False*,  
*\*\*body*)

Bases: *pandagg.node.mapping.abstract.Field*

```
KEY = 'text'
```

**class** pandagg.node.mapping.field\_datatypes.**TokenCount** (*name*, *depth=0*,  
*is\_subfield=False*, *\*\*body*)

Bases: *pandagg.node.mapping.abstract.Field*

To count the number of tokens in a string

```
KEY = 'token_count'
```

### pandagg.node.mapping.meta\_fields module

**class** pandagg.node.mapping.meta\_fields.**FieldNames** (*name*, *depth=0*, *is\_subfield=False*,  
*\*\*body*)

Bases: *pandagg.node.mapping.abstract.Field*

All fields in the document which contain non-null values.

```
KEY = '_field_names'
```

**class** pandagg.node.mapping.meta\_fields.**Id** (*name*, *depth=0*, *is\_subfield=False*, *\*\*body*)

Bases: *pandagg.node.mapping.abstract.Field*

The document's ID.

```
KEY = '_id'
```

**class** pandagg.node.mapping.meta\_fields.**Ignored** (*name*, *depth=0*, *is\_subfield=False*,  
*\*\*body*)

Bases: *pandagg.node.mapping.abstract.Field*

All fields in the document that have been ignored at index time because of ignore\_malformed.

```
KEY = '_ignored'
```

**class** pandagg.node.mapping.meta\_fields.**Index** (*name*, *depth=0*, *is\_subfield=False*,  
*\*\*body*)

Bases: *pandagg.node.mapping.abstract.Field*

The index to which the document belongs.

```
KEY = '_index'
```

**class** pandagg.node.mapping.meta\_fields.**Meta** (*name*, *depth=0*, *is\_subfield=False*, *\*\*body*)

Bases: *pandagg.node.mapping.abstract.Field*

Application specific metadata.

```
KEY = '_meta'
```

```
class pandagg.node.mapping.meta_fields.Routing(name, depth=0, is_subfield=False,
                                              **body)
```

Bases: *pandagg.node.mapping.abstract.Field*

A custom routing value which routes a document to a particular shard.

```
KEY = '_routing'
```

```
class pandagg.node.mapping.meta_fields.Size(name, depth=0, is_subfield=False, **body)
```

Bases: *pandagg.node.mapping.abstract.Field*

The size of the `_source` field in bytes, provided by the mapper-size plugin.

```
KEY = '_size'
```

```
class pandagg.node.mapping.meta_fields.Source(name, depth=0, is_subfield=False,
                                              **body)
```

Bases: *pandagg.node.mapping.abstract.Field*

The original JSON representing the body of the document.

```
KEY = '_source'
```

```
class pandagg.node.mapping.meta_fields.Type(name, depth=0, is_subfield=False, **body)
```

Bases: *pandagg.node.mapping.abstract.Field*

The document's mapping type.

```
KEY = '_type'
```

## Module contents

### pandagg.node.query package

#### Submodules

#### pandagg.node.query.abstract module

```
class pandagg.node.query.abstract.LeafQueryClause(_name=None, **body)
```

Bases: *pandagg.node.query.abstract.QueryClause*

```
class pandagg.node.query.abstract.MultiFieldsQueryClause(fields, _name=None,
                                                         **body)
```

Bases: *pandagg.node.query.abstract.LeafQueryClause*

```
tag
```

The readable node name for human. This attribute can be accessed and modified with `.` and `=` operator respectively.

```
class pandagg.node.query.abstract.QueryClause(_name=None, **body)
```

Bases: *pandagg.node.\_node.Node*

```
KEY = NotImplementedError()
```

```
NID_SIZE = 6
```

```
classmethod deserialize(**body)
```

```
name
```

```
serialize(named=False)
```

**tag**

The readable node name for human. This attribute can be accessed and modified with . and = operator respectively.

```
class pandagg.node.query.abstract.SingleFieldQueryClause (field, _name=None,
                                                         **body)
```

Bases: *pandagg.node.query.abstract.LeafQueryClause*

**FLAT = False**

**SHORT\_TAG = None**

**classmethod deserialize** (\*\*body)

**tag**

The readable node name for human. This attribute can be accessed and modified with . and = operator respectively.

## pandagg.node.query.compound module

```
class pandagg.node.query.compound.Bool (*args, **kwargs)
```

Bases: *pandagg.node.query.compound.CompoundClause*

**DEFAULT\_OPERATOR**

alias of pandagg.node.query.\_parameter\_clause.Must

**KEY = 'bool'**

**PARAMS\_WHITELIST = ['should', 'must', 'must\_not', 'filter', 'boost', 'minimum\_should\_must']**

```
class pandagg.node.query.compound.Boosting (*args, **kwargs)
```

Bases: *pandagg.node.query.compound.CompoundClause*

**DEFAULT\_OPERATOR**

alias of pandagg.node.query.\_parameter\_clause.Positive

**KEY = 'boosting'**

**PARAMS\_WHITELIST = ['positive', 'negative', 'negative\_boost']**

```
class pandagg.node.query.compound.CompoundClause (*args, **kwargs)
```

Bases: *pandagg.node.query.abstract.QueryClause*

Compound clauses can encapsulate other query clauses.

Note: the children attribute's only purpose is for initiation with the following syntax: >>> from pandagg.query import Bool, Term >>> query = Bool( >>> filter=Term(field='some\_path', value=3), >>> \_name='term\_agg', >>> ) Yet, the children attribute will then be reset to None to avoid confusion since the real hierarchy is stored in the bpointer/fpointer attributes inherited from treelib.Tree class.

**DEFAULT\_OPERATOR = NotImplementedError()**

**PARAMS\_WHITELIST = None**

**classmethod deserialize** (\*args, \*\*body)

**classmethod operator** (key)

**classmethod params** (parent\_only=False)

Return map of key -> params that handle children leaves.

```
class pandagg.node.query.compound.ConstantScore (*args, **kwargs)
```

Bases: *pandagg.node.query.compound.CompoundClause*

```

DEFAULT_OPERATOR
    alias of pandagg.node.query._parameter_clause.Filter

KEY = 'constant_score'

PARAMS_WHITELIST = ['filter', 'boost']

class pandagg.node.query.compound.DisMax(*args, **kwargs)
    Bases: pandagg.node.query.compound.CompoundClause

    DEFAULT_OPERATOR
        alias of pandagg.node.query._parameter_clause.Queries

    KEY = 'dis_max'

    PARAMS_WHITELIST = ['queries', 'tie_breaker']

class pandagg.node.query.compound.FunctionScore(*args, **kwargs)
    Bases: pandagg.node.query.compound.CompoundClause

    DEFAULT_OPERATOR
        alias of pandagg.node.query._parameter_clause.QueryP

    KEY = 'function_score'

    PARAMS_WHITELIST = ['query', 'boost', 'random_score', 'boost_mode', 'functions', 'max_

```

## pandagg.node.query.deserializer module

```

pandagg.node.query.deserializer.deserialize_node(k, body, accept_leaf=True,
                                                    accept_compound=True, ac-
                                                    cept_param=True)

```

## pandagg.node.query.full\_text module

```

class pandagg.node.query.full_text.Common(field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'common'

class pandagg.node.query.full_text.Intervals(field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'intervals'

class pandagg.node.query.full_text.Match(field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'match'

    SHORT_TAG = 'query'

class pandagg.node.query.full_text.MatchBoolPrefix(field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'match_bool_prefix'

    SHORT_TAG = 'query'

class pandagg.node.query.full_text.MatchPhrase(field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

```

```
KEY = 'match_phrase'
SHORT_TAG = 'query'

class pandagg.node.query.full_text.MatchPhrasePrefix (field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'match_phrase_prefix'
    SHORT_TAG = 'query'

class pandagg.node.query.full_text.MultiMatch (fields, _name=None, **body)
    Bases: pandagg.node.query.abstract.MultiFieldsQueryClause

    KEY = 'multi_match'

class pandagg.node.query.full_text.QueryString (_name=None, **body)
    Bases: pandagg.node.query.abstract.LeafQueryClause

    KEY = 'query_string'

class pandagg.node.query.full_text.SimpleQueryString (_name=None, **body)
    Bases: pandagg.node.query.abstract.LeafQueryClause

    KEY = 'simple_string'
```

### pandagg.node.query.geo module

```
class pandagg.node.query.geo.GeoBoundingBox (field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'geo_bounding_box'

class pandagg.node.query.geo.GeoDistance (field, location, distance, _name=None, **body)
    Bases: pandagg.node.query.abstract.LeafQueryClause

    KEY = 'geo_distance'

    classmethod deserialize (**body)

    tag
        The readable node name for human. This attribute can be accessed and modified with . and = operator respectively.

class pandagg.node.query.geo.GeoPolygone (field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'geo_polygon'

class pandagg.node.query.geo.GeoShape (field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'geo_shape'
```

### pandagg.node.query.joining module

```
class pandagg.node.query.joining.HasChild (*args, **kwargs)
    Bases: pandagg.node.query.compound.CompoundClause

    DEFAULT_OPERATOR
        alias of pandagg.node.query._parameter_clause.QueryP
```

```

    KEY = 'has_child'

    PARAMS_WHITELIST = ['query', 'type', 'max_children', 'min_children', 'score_mode', 'ig

class pandagg.node.query.joining.HasParent(*args, **kwargs)
    Bases: pandagg.node.query.compound.CompoundClause

    DEFAULT_OPERATOR
        alias of pandagg.node.query._parameter_clause.QueryP

    KEY = 'has_parent'

    PARAMS_WHITELIST = ['query', 'parent_type', 'score', 'ignore_unmapped']

class pandagg.node.query.joining.Nested(*args, **kwargs)
    Bases: pandagg.node.query.compound.CompoundClause

    DEFAULT_OPERATOR
        alias of pandagg.node.query._parameter_clause.QueryP

    KEY = 'nested'

    PARAMS_WHITELIST = ['path', 'query', 'score_mode', 'ignore_unmapped']

class pandagg.node.query.joining.ParentId(*args, **kwargs)
    Bases: pandagg.node.query.compound.CompoundClause

    KEY = 'parent_id'

```

### pandagg.node.query.shape module

```

class pandagg.node.query.shape.Shape(_name=None, **body)
    Bases: pandagg.node.query.abstract.LeafQueryClause

    KEY = 'shape'

```

### pandagg.node.query.span module

### pandagg.node.query.specialized module

```

class pandagg.node.query.specialized.DistanceFeature(field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    FLAT = True

    KEY = 'distance_feature'

class pandagg.node.query.specialized.MoreLikeThis(fields, _name=None, **body)
    Bases: pandagg.node.query.abstract.MultiFieldsQueryClause

    KEY = 'more_like_this'

class pandagg.node.query.specialized.Percolate(field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    FLAT = True

    KEY = 'percolate'

class pandagg.node.query.specialized.RankFeature(field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

```

```
FLAT = True
KEY = 'rank_feature'

class pandagg.node.query.specialized.Script(_name=None, **body)
    Bases: pandagg.node.query.abstract.LeafQueryClause
    KEY = 'script'

class pandagg.node.query.specialized.Wrapper(_name=None, **body)
    Bases: pandagg.node.query.abstract.LeafQueryClause
    KEY = 'wrapper'
```

### pandagg.node.query.specialized\_compound module

```
class pandagg.node.query.specialized_compound.PinnedQuery(*args, **kwargs)
    Bases: pandagg.node.query.compound.CompoundClause
    DEFAULT_OPERATOR
        alias of pandagg.node.query._parameter_clause.Organic
    KEY = 'pinned'
    PARAMS_WHITELIST = ['ids', 'organic']

class pandagg.node.query.specialized_compound.ScriptScore(*args, **kwargs)
    Bases: pandagg.node.query.compound.CompoundClause
    DEFAULT_OPERATOR
        alias of pandagg.node.query._parameter_clause.QueryP
    KEY = 'script_score'
    PARAMS_WHITELIST = ['query', 'script', 'min_score']
```

### pandagg.node.query.term\_level module

```
class pandagg.node.query.term_level.Exists(field, _name=None)
    Bases: pandagg.node.query.abstract.LeafQueryClause
    KEY = 'exists'
    tag
        The readable node name for human. This attribute can be accessed and modified with . and = operator respectively.

class pandagg.node.query.term_level.Fuzzy(field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause
    KEY = 'fuzzy'

class pandagg.node.query.term_level.Ids(values, _name=None)
    Bases: pandagg.node.query.abstract.LeafQueryClause
    KEY = 'ids'
    serialize (named=False)
    tag
        The readable node name for human. This attribute can be accessed and modified with . and = operator respectively.
```



```

class pandagg.node.query.term_level.Prefix (field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'prefix'

class pandagg.node.query.term_level.Range (field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'range'

class pandagg.node.query.term_level.Regexp (field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'regexp'

class pandagg.node.query.term_level.Term (field, value, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'term'

    SHORT_TAG = 'value'

class pandagg.node.query.term_level.Terms (field, terms, _name=None, **body)
    Bases: pandagg.node.query.abstract.LeafQueryClause

    KEY = 'terms'

    classmethod deserialize (**body)

    tag
        The readable node name for human. This attribute can be accessed and modified with . and = operator
        respectively.

class pandagg.node.query.term_level.TermsSet (field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'terms_set'

class pandagg.node.query.term_level.Type (field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'type'

class pandagg.node.query.term_level.Wildcard (field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'wildcard'

```

## Module contents

### pandagg.node.response package

#### Submodules

#### pandagg.node.response.bucket module

```

class pandagg.node.response.bucket.Bucket (depth, value, key=None, level=None)
    Bases: pandagg.node._node.Node

    ROOT_NAME = 'root'

```

**attr\_name**

Determine under which attribute name the bucket will be available in response tree. Dots are replaced by `_` characters so that they don't prevent from accessing as attribute.

Resulting attribute unfit for python attribute name syntax is still possible and will be accessible through item access (dict like), see more in 'utils.Obj' for more details.

**display\_name****display\_name\_with\_value****Module contents****5.1.2.2 Submodules****pandagg.node.mixins module**

```
class pandagg.node.mixins.FieldValidityMixin
```

Bases: `object`

```
BLACKLISTED_MAPPING_TYPES = None
```

```
WHITELISTED_MAPPING_TYPES = None
```

```
classmethod valid_on_field_type(field_type)
```

**pandagg.node.types module****5.1.2.3 Module contents****5.1.3 pandagg.tree package****5.1.3.1 Submodules****pandagg.tree.agg module**

```
class pandagg.tree.agg.Agg(from_=None, mapping=None, identifier=None, client=None,  
                          query=None, index_name=None)
```

Bases: `pandagg.tree._tree.Tree`

Tree combination of aggregation nodes.

Mapping declaration is optional, but doing so validates aggregation validity.

```
DEFAULT_OUTPUT = 'dataframe'
```

```
add_node (node, pid=None)
```

If mapping is provided, nested and outnested are automatically applied.

```
agg (arg, insert_below=None, **kwargs)
```

Arrange passed aggregations in *arg* arguments “horizontally”.

Those will be placed under the *insert\_below* aggregation clause id if provided, else under the deepest linear bucket aggregation if there is no ambiguity: OK: A—> B -> C -> arg KO: A—> B

└─> C

*arg* argument accepts single occurrence or sequence of following formats: - string (for terms agg concise declaration) - regular Elasticsearch dict syntax - AggNode instance (for instance Terms, Filters etc)

#### Parameters

- **arg** – aggregation(s) clauses to insert “horizontally”
- **insert\_below** – parent aggregation id under which these aggregations should be placed
- **kwargs** – agg body arguments when using “string” syntax for terms aggregation

**Return type** [\*pandagg.agg.Agg\*](#)

**applied\_nested\_path\_at\_node** (*nid*)

**bind** (*client*, *index\_name=None*)

**deepest\_linear\_bucket\_agg**

Return deepest bucket aggregation node ([\*pandagg.nodes.abstract.BucketAggNode\*](#)) of that aggregation that neither has siblings, nor has an ancestor with siblings.

**classmethod deserialize** (*from\_*)

**execute** (*index=None*, *output='dataframe'*, *\*\*kwargs*)

**groupby** (*by*, *insert\_below=None*, *insert\_above=None*, *\*\*kwargs*)

Arrange passed aggregations in *by* arguments “vertically” (nested manner), above or below another agg clause.

Given the initial aggregation:  $A \rightarrow B \hookrightarrow C$

If *insert\_below* = ‘A’:  $A \rightarrow by \rightarrow B$

$\hookrightarrow C$

If *insert\_above* = ‘B’:  $A \rightarrow by \rightarrow B \hookrightarrow C$

*by* argument accepts single occurrence or sequence of following formats: - string (for terms agg concise declaration) - regular Elasticsearch dict syntax - AggNode instance (for instance Terms, Filters etc)

If *insert\_below* nor *insert\_above* is provided by will be placed between the the deepest linear bucket aggregation if there is no ambiguity, and its children:  $A \rightarrow B$  : OK generates  $A \rightarrow B \rightarrow C \rightarrow by$

$A \rightarrow B$  : KO, ambiguous, must precise either A, B or C  $\hookrightarrow C$

#### Parameters

- **by** – aggregation(s) clauses to insert “vertically”
- **insert\_below** – parent aggregation id under which these aggregations should be placed
- **insert\_above** – aggregation id above which these aggregations should be placed
- **kwargs** – agg body arguments when using “string” syntax for terms aggregation

**Return type** [\*pandagg.agg.Agg\*](#)

**node\_class**

alias of [\*pandagg.node.agg.abstract.AggNode\*](#)

**paste** (*nid*, *new\_tree*, *deep=False*)

Pastes a tree handling nested implications if mapping is provided. The provided tree should be validated beforehand.

**query** (*query*, *validate=False*, *\*\*kwargs*)

**query\_dict** (*from\_=None, depth=None, with\_name=True*)

**serialize\_response** (*aggs, output, \*\*kwargs*)

**set\_mapping** (*mapping*)

**validate\_tree** (*exc=False*)

Validate tree definition against defined mapping. :param exc: if set to True, will raise exception if tree is invalid :return: boolean

## pandagg.tree.mapping module

**class** pandagg.tree.mapping.**Mapping** (*from\_=None, identifier=None, properties=None, dynamic=False*)

Bases: pandagg.tree.\_tree.Tree

**contains** (*nid*)

Check if the tree contains node of given id

**classmethod** **deserialize** (*from\_, depth=0*)

**list\_nesteds\_at\_field** (*field\_path*)

**mapping\_type\_of\_field** (*field\_path*)

**nested\_at\_field** (*field\_path*)

**node\_class**

alias of `pandagg.node.mapping.abstract.Field`

**node\_path** (*nid*)

**serialize** ()

**show** (*data\_property='pretty', \*\*kwargs*)

Print the tree structure in hierarchy style.

You have three ways to output your tree data, i.e., stdout with `show()`, plain text file with `save2file()`, and json string with `to_json()`. The former two use the same backend to generate a string of tree structure in a text graph.

- Version >= 1.2.7a\*: you can also specify the `line_type` parameter, such as 'ascii' (default), 'ascii-ex', 'ascii-exr', 'ascii-em', 'ascii-emv', 'ascii-emh') to the change graphical form.

### Parameters

- **nid** – the reference node to start expanding.
- **level** – the node level in the tree (root as level 0).
- **idhidden** – whether hiding the node ID when printing.
- **filter** – the function of one variable to act on the Node object. When this parameter is specified, the traversing will not continue to following children of node whose condition does not pass the filter.
- **key** – the key param for sorting Node objects in the same level.
- **reverse** – the `reverse` param for sorting Node objects in the same level.
- **line\_type** –
- **data\_property** – the property on the node data object to be printed.

**Returns** None

**validate\_agg\_node** (*agg\_node*, *exc=True*)

Ensure if node has field or path that it exists in mapping, and that required aggregation type if allowed on this kind of field. :param *agg\_node*: AggNode you want to validate on this mapping :param *exc*: boolean, if set to True raise exception if invalid :rtype: boolean

## pandagg.tree.query module

**class** pandagg.tree.query.**Query** (*from\_=None*, *mapping=None*, *identifier=None*, *client=None*, *index\_name=None*)

Bases: pandagg.tree.\_tree.Tree

Tree combination of query nodes.

Mapping declaration is optional, but doing so validates query validity.

**add\_node** (*node*, *pid=None*)

Add a new node object to the tree and make the parent as the root by default.

The ‘node’ parameter refers to an instance of Class::Node.

**bind** (*client*, *index\_name=None*)

**bool** (*\*args*, *\*\*kwargs*)

**boost** (*\*args*, *\*\*kwargs*)

**constant\_score** (*\*args*, *\*\*kwargs*)

**classmethod deserialize** (*from\_*)

**dis\_max** (*\*args*, *\*\*kwargs*)

**execute** (*index=None*, *\*\*kwargs*)

**filter** (*\*args*, *\*\*kwargs*)

**function\_score** (*\*args*, *\*\*kwargs*)

**has\_child** (*\*args*, *\*\*kwargs*)

**has\_parent** (*\*args*, *\*\*kwargs*)

**must** (*\*args*, *\*\*kwargs*)

**must\_not** (*\*args*, *\*\*kwargs*)

**nested** (*\*args*, *\*\*kwargs*)

**node\_class**

alias of `pandagg.node.query.abstract.QueryClause`

**parent\_id** (*\*args*, *\*\*kwargs*)

**pinned\_query** (*\*args*, *\*\*kwargs*)

**query** (*q*, *parent=None*, *child=None*, *parent\_param=None*, *child\_param=None*, *mode='add'*)

Place query below a given parent.

**query\_dict** (*from\_=None*, *named=False*)

Return None if no query clause.

**script\_score** (*\*args*, *\*\*kwargs*)

**set\_mapping** (*mapping*)

**should** (*\*args*, *\*\*kwargs*)

## pandagg.tree.response module

**class** pandagg.tree.response.**ResponseTree** (*agg\_tree, identifier=None*)

Bases: pandagg.tree.\_tree.Tree

Tree representation of an ES response. ES response format is determined by the aggregation query.

**bucket\_properties** (*bucket, properties=None, end\_level=None, depth=None*)

Recursive method returning a given bucket's properties in the form of an ordered dictionary. Travel from current bucket through all ancestors until reaching root. :param bucket: instance of pandagg.buckets.buckets.Bucket :param properties: OrderedDict accumulator of 'level' -> 'key' :param end\_level: optional parameter to specify until which level properties are fetched :param depth: optional parameter to specify a limit number of levels which are fetched :return: OrderedDict of structure 'level' -> 'key'

**get\_bucket\_filter** (*nid*)

Build query filtering documents belonging to that bucket. Suppose the following configuration:

**Base** <- filter on **base** | — Nested\_A no filter on A (nested still must be applied for children) | | — Sub-Nested A1 | — SubNested A2 <- filter on A2 | — Nested\_B <- filter on B

**parse\_aggregation** (*raw\_response*)

Build response tree from ES response :param raw\_response: ES aggregation response :return: self

Note: if the root aggregation node can generate multiple buckets, a response root is crafted to avoid having multiple roots.

**show** (*data\_property='pretty', \*\*kwargs*)

Print the tree structure in hierarchy style.

You have three ways to output your tree data, i.e., stdout with `show()`, plain text file with `save2file()`, and json string with `to_json()`. The former two use the same backend to generate a string of tree structure in a text graph.

- Version >= 1.2.7a\*: you can also specify the `line_type` parameter, such as 'ascii' (default), 'ascii-ex', 'ascii-exr', 'ascii-em', 'ascii-emv', 'ascii-emh') to the change graphical form.

### Parameters

- **nid** – the reference node to start expanding.
- **level** – the node level in the tree (root as level 0).
- **idhidden** – whether hiding the node ID when printing.
- **filter** – the function of one variable to act on the Node object. When this parameter is specified, the traversing will not continue to following children of node whose condition does not pass the filter.
- **key** – the key param for sorting Node objects in the same level.
- **reverse** – the reverse param for sorting Node objects in the same level.
- **line\_type** –
- **data\_property** – the property on the node data object to be printed.

**Returns** None

### 5.1.3.2 Module contents

## 5.2 Submodules

### 5.2.1 pandagg.agg module

**class** `pandagg.agg.Agg` (*from\_=None, mapping=None, identifier=None, client=None, query=None, index\_name=None*)

Bases: `pandagg.tree._tree.Tree`

Tree combination of aggregation nodes.

Mapping declaration is optional, but doing so validates aggregation validity.

**DEFAULT\_OUTPUT** = 'dataframe'

**add\_node** (*node, pid=None*)

If mapping is provided, nested and outnested are automatically applied.

**agg** (*arg, insert\_below=None, \*\*kwargs*)

Arrange passed aggregations in *arg* arguments “horizontally”.

Those will be placed under the *insert\_below* aggregation clause id if provided, else under the deepest linear bucket aggregation if there is no ambiguity: OK:  $A \rightarrow B \rightarrow C \rightarrow \text{arg}$  KO:  $A \rightarrow B$

$\hookrightarrow C$

*arg* argument accepts single occurrence or sequence of following formats: - string (for terms agg concise declaration) - regular Elasticsearch dict syntax - `AggNode` instance (for instance Terms, Filters etc)

#### Parameters

- **arg** – aggregation(s) clauses to insert “horizontally”
- **insert\_below** – parent aggregation id under which these aggregations should be placed
- **kwargs** – agg body arguments when using “string” syntax for terms aggregation

**Return type** `pandagg.agg.Agg`

**applied\_nested\_path\_at\_node** (*nid*)

**bind** (*client, index\_name=None*)

**deepest\_linear\_bucket\_agg**

Return deepest bucket aggregation node (`pandagg.nodes.abstract.BucketAggNode`) of that aggregation that neither has siblings, nor has an ancestor with siblings.

**classmethod deserialize** (*from\_*)

**execute** (*index=None, output='dataframe', \*\*kwargs*)

**groupby** (*by, insert\_below=None, insert\_above=None, \*\*kwargs*)

Arrange passed aggregations in *by* arguments “vertically” (nested manner), above or below another agg clause.

Given the initial aggregation:  $A \rightarrow B \hookrightarrow C$

If *insert\_below* = 'A':  $A \rightarrow \text{by} \rightarrow B$

$\hookrightarrow C$

If *insert\_above* = 'B':  $A \longrightarrow \text{by} \longrightarrow B \longmapsto C$

*by* argument accepts single occurrence or sequence of following formats: - string (for terms agg concise declaration) - regular Elasticsearch dict syntax - *AggNode* instance (for instance Terms, Filters etc)

If *insert\_below* nor *insert\_above* is provided *by* will be placed between the the deepest linear bucket aggregation if there is no ambiguity, and its children:  $A \longrightarrow B$  : OK generates  $A \longrightarrow B \rightarrow C \rightarrow \text{by}$

$A \longrightarrow B$  : KO, ambiguous, must precise either A, B or C  $\longmapsto C$

#### Parameters

- **by** – aggregation(s) clauses to insert “vertically”
- **insert\_below** – parent aggregation id under which these aggregations should be placed
- **insert\_above** – aggregation id above which these aggregations should be placed
- **kwargs** – agg body arguments when using “string” syntax for terms aggregation

Return type *pandagg.agg.Agg*

#### **node\_class**

alias of *pandagg.node.agg.abstract.AggNode*

#### **paste** (*nid*, *new\_tree*, *deep=False*)

Pastes a tree handling nested implications if mapping is provided. The provided tree should be validated beforehand.

#### **query** (*query*, *validate=False*, *\*\*kwargs*)

#### **query\_dict** (*from\_=None*, *depth=None*, *with\_name=True*)

#### **serialize\_response** (*aggs*, *output*, *\*\*kwargs*)

#### **set\_mapping** (*mapping*)

#### **validate\_tree** (*exc=False*)

Validate tree definition against defined mapping. :param exc: if set to True, will raise exception if tree is invalid :return: boolean

#### **class** *pandagg.agg.MatchAll* (*name*, *meta=None*, *aggs=None*)

Bases: *pandagg.node.agg.bucket.Filter*

#### **class** *pandagg.agg.Terms* (*name*, *field*, *missing=None*, *size=None*, *aggs=None*, *meta=None*, *\*\*body*)

Bases: *pandagg.node.agg.abstract.MultipleBucketAgg*

Terms aggregation.

**BLACKLISTED\_MAPPING\_TYPES** = []

**KEY** = 'terms'

**VALUE\_ATTRS** = ['doc\_count', 'doc\_count\_error\_upper\_bound', 'sum\_other\_doc\_count']

#### **get\_filter** (*key*)

Provide filter to get documents belonging to document of given key.

#### **class** *pandagg.agg.Filters* (*name*, *filters*, *other\_bucket=False*, *other\_bucket\_key=None*, *meta=None*, *aggs=None*, *\*\*body*)

Bases: *pandagg.node.agg.abstract.MultipleBucketAgg*

**DEFAULT\_OTHER\_KEY** = '\_other\_'

**IMPLICIT\_KEYED** = True

**KEY** = 'filters'



```

    VALUE_ATTRS = ['doc_count']

    get_filter(key)
        Provide filter to get documents belonging to document of given key.

class pandagg.agg.Histogram(name, field, interval, meta=None, aggs=None, **body)
    Bases: pandagg.node.agg.abstract.MultipleBucketAgg

    KEY = 'histogram'

    VALUE_ATTRS = ['doc_count']

    WHITELISTED_MAPPING_TYPES = ['long', 'integer', 'short', 'byte', 'double', 'float', 'h

    get_filter(key)
        Provide filter to get documents belonging to document of given key.

class pandagg.agg.DateHistogram(name, field, interval=None, calendar_interval=None,
                                fixed_interval=None, meta=None, keyed=False,
                                key_as_string=True, aggs=None, **body)
    Bases: pandagg.node.agg.abstract.MultipleBucketAgg

    KEY = 'date_histogram'

    VALUE_ATTRS = ['doc_count']

    WHITELISTED_MAPPING_TYPES = ['date']

    get_filter(key)
        Provide filter to get documents belonging to document of given key.

class pandagg.agg.Range(name, field, ranges, keyed=False, meta=None, aggs=None, **body)
    Bases: pandagg.node.agg.abstract.MultipleBucketAgg

    KEY = 'range'

    KEY_SEP = '-'

    VALUE_ATTRS = ['doc_count']

    WHITELISTED_MAPPING_TYPES = ['long', 'integer', 'short', 'byte', 'double', 'float', 'h

    from_key

    get_filter(key)
        Provide filter to get documents belonging to document of given key.

    to_key

class pandagg.agg.Global(name, meta=None, aggs=None)
    Bases: pandagg.node.agg.abstract.UniqueBucketAgg

    KEY = 'global'

    VALUE_ATTRS = ['doc_count']

    get_filter(key)
        Provide filter to get documents belonging to document of given key.

class pandagg.agg.Filter(name, filter, meta=None, aggs=None, **body)
    Bases: pandagg.node.agg.abstract.UniqueBucketAgg

    KEY = 'filter'

    VALUE_ATTRS = ['doc_count']

    get_filter(key)
        Provide filter to get documents belonging to document of given key.

```

```
class pandagg.agg.Nested(name, path, meta=None, aggs=None)
    Bases: pandagg.node.agg.abstract.UniqueBucketAgg

    KEY = 'nested'

    VALUE_ATTRS = ['doc_count']

    WHITELISTED_MAPPING_TYPES = ['nested']

    get_filter(key)
        Provide filter to get documents belonging to document of given key.

class pandagg.agg.ReverseNested(name, path=None, meta=None, aggs=None, **body)
    Bases: pandagg.node.agg.abstract.UniqueBucketAgg

    KEY = 'reverse_nested'

    VALUE_ATTRS = ['doc_count']

    WHITELISTED_MAPPING_TYPES = ['nested']

    get_filter(key)
        Provide filter to get documents belonging to document of given key.

class pandagg.agg.Avg(name, meta=None, **body)
    Bases: pandagg.node.agg.abstract.FieldOrScriptMetricAgg

    KEY = 'avg'

    VALUE_ATTRS = ['value']

    WHITELISTED_MAPPING_TYPES = ['long', 'integer', 'short', 'byte', 'double', 'float', 'h

class pandagg.agg.Max(name, meta=None, **body)
    Bases: pandagg.node.agg.abstract.FieldOrScriptMetricAgg

    KEY = 'max'

    VALUE_ATTRS = ['value']

    WHITELISTED_MAPPING_TYPES = ['long', 'integer', 'short', 'byte', 'double', 'float', 'h

class pandagg.agg.Sum(name, meta=None, **body)
    Bases: pandagg.node.agg.abstract.FieldOrScriptMetricAgg

    KEY = 'sum'

    VALUE_ATTRS = ['value']

    WHITELISTED_MAPPING_TYPES = ['long', 'integer', 'short', 'byte', 'double', 'float', 'h

class pandagg.agg.Min(name, meta=None, **body)
    Bases: pandagg.node.agg.abstract.FieldOrScriptMetricAgg

    KEY = 'min'

    VALUE_ATTRS = ['value']

    WHITELISTED_MAPPING_TYPES = ['long', 'integer', 'short', 'byte', 'double', 'float', 'h

class pandagg.agg.Cardinality(name, meta=None, **body)
    Bases: pandagg.node.agg.abstract.FieldOrScriptMetricAgg

    KEY = 'cardinality'

    VALUE_ATTRS = ['value']
```

```

class pandagg.agg.Stats(name, meta=None, **body)
    Bases: pandagg.node.agg.abstract.FieldOrScriptMetricAgg

    KEY = 'stats'

    VALUE_ATTRS = ['count', 'min', 'max', 'avg', 'sum']

    WHITELISTED_MAPPING_TYPES = ['long', 'integer', 'short', 'byte', 'double', 'float', 'h

class pandagg.agg.ExtendedStats(name, meta=None, **body)
    Bases: pandagg.node.agg.abstract.FieldOrScriptMetricAgg

    KEY = 'extended_stats'

    VALUE_ATTRS = ['count', 'min', 'max', 'avg', 'sum', 'sum_of_squares', 'variance', 'std

    WHITELISTED_MAPPING_TYPES = ['long', 'integer', 'short', 'byte', 'double', 'float', 'h

class pandagg.agg.Percentiles(name, meta=None, **body)
    Bases: pandagg.node.agg.abstract.FieldOrScriptMetricAgg

    Percents body argument can be passed to specify which percentiles to fetch.

    KEY = 'percentiles'

    VALUE_ATTRS = ['values']

    WHITELISTED_MAPPING_TYPES = ['long', 'integer', 'short', 'byte', 'double', 'float', 'h

class pandagg.agg.PercentileRanks(name, field, values, meta=None, **body)
    Bases: pandagg.node.agg.abstract.FieldOrScriptMetricAgg

    KEY = 'percentile_ranks'

    VALUE_ATTRS = ['values']

    WHITELISTED_MAPPING_TYPES = ['long', 'integer', 'short', 'byte', 'double', 'float', 'h

class pandagg.agg.GeoBound(name, meta=None, **body)
    Bases: pandagg.node.agg.abstract.FieldOrScriptMetricAgg

    KEY = 'geo_bounds'

    VALUE_ATTRS = ['bounds']

    WHITELISTED_MAPPING_TYPES = ['geo_point']

class pandagg.agg.GeoCentroid(name, meta=None, **body)
    Bases: pandagg.node.agg.abstract.FieldOrScriptMetricAgg

    KEY = 'geo_centroid'

    VALUE_ATTRS = ['location']

    WHITELISTED_MAPPING_TYPES = ['geo_point']

class pandagg.agg.TopHits(name, meta=None, **body)
    Bases: pandagg.node.agg.abstract.MetricAgg

    KEY = 'top_hits'

    VALUE_ATTRS = ['hits']

class pandagg.agg.ValueCount(name, meta=None, **body)
    Bases: pandagg.node.agg.abstract.FieldOrScriptMetricAgg

    BLACKLISTED_MAPPING_TYPES = []

    KEY = 'value_count'

```

```
    VALUE_ATTRS = ['value']

class pandagg.agg.AvgBucket(name, buckets_path, gap_policy=None, meta=None, aggs=None,
                           **body)
    Bases: pandagg.node.agg.abstract.Pipeline
    KEY = 'avg_bucket'
    VALUE_ATTRS = ['value']

class pandagg.agg.Derivative(name, buckets_path, gap_policy=None, meta=None, aggs=None,
                             **body)
    Bases: pandagg.node.agg.abstract.Pipeline
    KEY = 'derivative'
    VALUE_ATTRS = ['value']

class pandagg.agg.MaxBucket(name, buckets_path, gap_policy=None, meta=None, aggs=None,
                            **body)
    Bases: pandagg.node.agg.abstract.Pipeline
    KEY = 'max_bucket'
    VALUE_ATTRS = ['value']

class pandagg.agg.MinBucket(name, buckets_path, gap_policy=None, meta=None, aggs=None,
                             **body)
    Bases: pandagg.node.agg.abstract.Pipeline
    KEY = 'min_bucket'
    VALUE_ATTRS = ['value']

class pandagg.agg.SumBucket(name, buckets_path, gap_policy=None, meta=None, aggs=None,
                             **body)
    Bases: pandagg.node.agg.abstract.Pipeline
    KEY = 'sum_bucket'
    VALUE_ATTRS = ['value']

class pandagg.agg.StatsBucket(name, buckets_path, gap_policy=None, meta=None, aggs=None,
                              **body)
    Bases: pandagg.node.agg.abstract.Pipeline
    KEY = 'stats_bucket'
    VALUE_ATTRS = ['count', 'min', 'max', 'avg', 'sum']

class pandagg.agg.ExtendedStatsBucket(name, buckets_path, gap_policy=None, meta=None,
                                       aggs=None, **body)
    Bases: pandagg.node.agg.abstract.Pipeline
    KEY = 'extended_stats_bucket'
    VALUE_ATTRS = ['count', 'min', 'max', 'avg', 'sum', 'sum_of_squares', 'variance', 'std']

class pandagg.agg.PercentilesBucket(name, buckets_path, gap_policy=None, meta=None,
                                      aggs=None, **body)
    Bases: pandagg.node.agg.abstract.Pipeline
    KEY = 'percentiles_bucket'
    VALUE_ATTRS = ['values']
```

```

class pandagg.agg.MovingAvg(name, buckets_path, gap_policy=None, meta=None, aggs=None,
                             **body)
    Bases: pandagg.node.agg.abstract.Pipeline
    KEY = 'moving_avg'
    VALUE_ATTRS = ['value']

class pandagg.agg.CumulativeSum(name, buckets_path, gap_policy=None, meta=None,
                                 aggs=None, **body)
    Bases: pandagg.node.agg.abstract.Pipeline
    KEY = 'cumulative_sum'
    VALUE_ATTRS = ['value']

class pandagg.agg.BucketScript(name, script, buckets_path, gap_policy=None, meta=None,
                                aggs=None, **body)
    Bases: pandagg.node.agg.abstract.ScriptPipeline
    KEY = 'bucket_script'
    VALUE_ATTRS = ['value']

class pandagg.agg.BucketSelector(name, script, buckets_path, gap_policy=None, meta=None,
                                  aggs=None, **body)
    Bases: pandagg.node.agg.abstract.ScriptPipeline
    KEY = 'bucket_selector'
    VALUE_ATTRS = None

class pandagg.agg.BucketSort(name, script, buckets_path, gap_policy=None, meta=None,
                              aggs=None, **body)
    Bases: pandagg.node.agg.abstract.ScriptPipeline
    KEY = 'bucket_sort'
    VALUE_ATTRS = None

class pandagg.agg.SerialDiff(name, buckets_path, gap_policy=None, meta=None, aggs=None,
                              **body)
    Bases: pandagg.node.agg.abstract.Pipeline
    KEY = 'serial_diff'
    VALUE_ATTRS = ['value']

```

## 5.2.2 pandagg.client module

```

class pandagg.client.Elasticsearch(hosts=None, transport_class=<class 'elastic-
                                search.transport.Transport'>, **kwargs)
    Bases: elasticsearch.client.Elasticsearch
    fetch_indices(index='*')

```

**Parameters** **index** – Comma-separated list or wildcard expression of index names used to limit the request.

## 5.2.3 pandagg.exceptions module

```

exception pandagg.exceptions.AbsentMappingFieldError
    Bases: pandagg.exceptions.MappingError

```

Field is not present in mapping.

**exception** `pandagg.exceptions.InvalidAggregation`

Bases: `Exception`

Wrong aggregation definition

**exception** `pandagg.exceptions.InvalidOperationMappingFieldError`

Bases: `pandagg.exceptions.MappingError`

Invalid aggregation type on this mapping field.

**exception** `pandagg.exceptions.MappingError`

Bases: `Exception`

Basic Mapping Error

**exception** `pandagg.exceptions.VersionIncompatibilityError`

Bases: `Exception`

Pandagg is not compatible with this ElasticSearch version.

## 5.2.4 pandagg.mapping module

**class** `pandagg.mapping.Mapping` (*from\_=None, identifier=None, properties=None, dynamic=False*)

Bases: `pandagg.tree._tree.Tree`

**contains** (*nid*)

Check if the tree contains node of given id

**classmethod** **deserialize** (*from\_, depth=0*)

**list\_nesteds\_at\_field** (*field\_path*)

**mapping\_type\_of\_field** (*field\_path*)

**nested\_at\_field** (*field\_path*)

**node\_class**

alias of `pandagg.node.mapping.abstract.Field`

**node\_path** (*nid*)

**serialize** ()

**show** (*data\_property='pretty', \*\*kwargs*)

Print the tree structure in hierarchy style.

You have three ways to output your tree data, i.e., stdout with `show()`, plain text file with `save2file()`, and json string with `to_json()`. The former two use the same backend to generate a string of tree structure in a text graph.

- Version `>= 1.2.7a*`: you can also specify the `line_type` parameter, such as `'ascii'` (default), `'ascii-ex'`, `'ascii-exr'`, `'ascii-em'`, `'ascii-emv'`, `'ascii-emh'`) to the change graphical form.

### Parameters

- **nid** – the reference node to start expanding.
- **level** – the node level in the tree (root as level 0).
- **idhidden** – whether hiding the node ID when printing.

- **filter** – the function of one variable to act on the `Node` object. When this parameter is specified, the traversing will not continue to following children of node whose condition does not pass the filter.
- **key** – the key param for sorting `Node` objects in the same level.
- **reverse** – the `reverse` param for sorting `Node` objects in the same level.
- **line\_type** –
- **data\_property** – the property on the node data object to be printed.

Returns `None`

**validate\_agg\_node** (*agg\_node*, *exc=True*)

Ensure if node has field or path that it exists in mapping, and that required aggregation type if allowed on this kind of field. :param *agg\_node*: `AggNode` you want to validate on this mapping :param *exc*: boolean, if set to `True` raise exception if invalid :rtype: boolean

**class** `pandagg.mapping.IMapping` (*from\_=None*, *properties=None*, *dynamic=False*, *client=None*,  
*root\_path=None*, *depth=1*, *initial\_tree=None*, *index\_name=None*)

Bases: `pandagg.interactive.abstract.TreeBasedObj`

Interactive wrapper upon mapping tree.

**class** `pandagg.mapping.Text` (*name*, *depth=0*, *is\_subfield=False*, *\*\*body*)

Bases: `pandagg.node.mapping.abstract.Field`

**KEY** = 'text'

**class** `pandagg.mapping.Keyword` (*name*, *depth=0*, *is\_subfield=False*, *\*\*body*)

Bases: `pandagg.node.mapping.abstract.Field`

**KEY** = 'keyword'

**class** `pandagg.mapping.Long` (*name*, *depth=0*, *is\_subfield=False*, *\*\*body*)

Bases: `pandagg.node.mapping.abstract.Field`

**KEY** = 'long'

**class** `pandagg.mapping.Integer` (*name*, *depth=0*, *is\_subfield=False*, *\*\*body*)

Bases: `pandagg.node.mapping.abstract.Field`

**KEY** = 'integer'

**class** `pandagg.mapping.Short` (*name*, *depth=0*, *is\_subfield=False*, *\*\*body*)

Bases: `pandagg.node.mapping.abstract.Field`

**KEY** = 'short'

**class** `pandagg.mapping.Byte` (*name*, *depth=0*, *is\_subfield=False*, *\*\*body*)

Bases: `pandagg.node.mapping.abstract.Field`

**KEY** = 'byte'

**class** `pandagg.mapping.Double` (*name*, *depth=0*, *is\_subfield=False*, *\*\*body*)

Bases: `pandagg.node.mapping.abstract.Field`

**KEY** = 'double'

**class** `pandagg.mapping.HalfFloat` (*name*, *depth=0*, *is\_subfield=False*, *\*\*body*)

Bases: `pandagg.node.mapping.abstract.Field`

**KEY** = 'half\_float'

```
class pandagg.mapping.ScaledFloat (name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    KEY = 'scaled_float'

class pandagg.mapping.Date (name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    KEY = 'date'

class pandagg.mapping.DateNanos (name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    KEY = 'date_nanos'

class pandagg.mapping.Boolean (name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    KEY = 'boolean'

class pandagg.mapping.Binary (name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    KEY = 'binary'

class pandagg.mapping.IntegerRange (name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    KEY = 'integer_range'

class pandagg.mapping.Float (name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    KEY = 'float'

class pandagg.mapping.FloatRange (name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    KEY = 'float_range'

class pandagg.mapping.LongRange (name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    KEY = 'long_range'

class pandagg.mapping.DoubleRange (name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    KEY = 'double_range'

class pandagg.mapping.DateRange (name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    KEY = 'date_range'

class pandagg.mapping.Object (name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    DISPLAY_PATTERN = ' {%s}'

    KEY = 'object'

class pandagg.mapping.Nested (name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    DISPLAY_PATTERN = ' [%s]'
```



```

    KEY = 'nested'

class pandagg.mapping.GeoPoint (name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    For lat/lon points

    KEY = 'geo_point'

class pandagg.mapping.GeoShape (name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    For complex shapes like polygons

    KEY = 'geo_shape'

class pandagg.mapping.IP (name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    for IPv4 and IPv6 addresses

    KEY = 'IP'

class pandagg.mapping.Completion (name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    To provide auto-complete suggestions

    KEY = 'completion'

class pandagg.mapping.TokenCount (name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    To count the number of tokens in a string

    KEY = 'token_count'

class pandagg.mapping.MapperMurMur3 (name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    To compute hashes of values at index-time and store them in the index

    KEY = 'murmur3'

class pandagg.mapping.MapperAnnotatedText (name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    To index text containing special markup (typically used for identifying named entities)

    KEY = 'annotated-text'

class pandagg.mapping.Percolator (name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    Accepts queries from the query-dsl

    KEY = 'percolator'

class pandagg.mapping.Join (name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    Defines parent/child relation for documents within the same index

    KEY = 'join'

class pandagg.mapping.RankFeature (name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

```

Record numeric feature to boost hits at query time.

**KEY = 'rank\_feature'**

**class** pandagg.mapping.**RankFeatures** (*name, depth=0, is\_subfield=False, \*\*body*)

Bases: *pandagg.node.mapping.abstract.Field*

Record numeric features to boost hits at query time.

**KEY = 'rank\_features'**

**class** pandagg.mapping.**DenseVector** (*name, depth=0, is\_subfield=False, \*\*body*)

Bases: *pandagg.node.mapping.abstract.Field*

Record dense vectors of float values.

**KEY = 'dense\_vector'**

**class** pandagg.mapping.**SparseVector** (*name, depth=0, is\_subfield=False, \*\*body*)

Bases: *pandagg.node.mapping.abstract.Field*

Record sparse vectors of float values.

**KEY = 'sparse\_vector'**

**class** pandagg.mapping.**SearchAsYouType** (*name, depth=0, is\_subfield=False, \*\*body*)

Bases: *pandagg.node.mapping.abstract.Field*

A text-like field optimized for queries to implement as-you-type completion

**KEY = 'search\_as\_you\_type'**

**class** pandagg.mapping.**Alias** (*name, depth=0, is\_subfield=False, \*\*body*)

Bases: *pandagg.node.mapping.abstract.Field*

Defines an alias to an existing field.

**KEY = 'alias'**

**class** pandagg.mapping.**Flattened** (*name, depth=0, is\_subfield=False, \*\*body*)

Bases: *pandagg.node.mapping.abstract.Field*

Allows an entire JSON object to be indexed as a single field.

**KEY = 'flattened'**

**class** pandagg.mapping.**Shape** (*name, depth=0, is\_subfield=False, \*\*body*)

Bases: *pandagg.node.mapping.abstract.Field*

For arbitrary cartesian geometries.

**KEY = 'shape'**

**class** pandagg.mapping.**Histogram** (*name, depth=0, is\_subfield=False, \*\*body*)

Bases: *pandagg.node.mapping.abstract.Field*

For pre-aggregated numerical values for percentiles aggregations.

**KEY = 'histogram'**

**class** pandagg.mapping.**Index** (*name, depth=0, is\_subfield=False, \*\*body*)

Bases: *pandagg.node.mapping.abstract.Field*

The index to which the document belongs.

**KEY = '\_index'**

```

class pandagg.mapping.Type(name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    The document's mapping type.

    KEY = '_type'

class pandagg.mapping.Id(name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    The document's ID.

    KEY = '_id'

class pandagg.mapping.Source(name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    The original JSON representing the body of the document.

    KEY = '_source'

class pandagg.mapping.Size(name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    The size of the _source field in bytes, provided by the mapper-size plugin.

    KEY = '_size'

class pandagg.mapping.FieldNames(name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    All fields in the document which contain non-null values.

    KEY = '_field_names'

class pandagg.mapping.Ignored(name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    All fields in the document that have been ignored at index time because of ignore_malformed.

    KEY = '_ignored'

class pandagg.mapping.Routing(name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    A custom routing value which routes a document to a particular shard.

    KEY = '_routing'

class pandagg.mapping.Meta(name, depth=0, is_subfield=False, **body)
    Bases: pandagg.node.mapping.abstract.Field

    Application specific metadata.

    KEY = '_meta'

```

### 5.2.5 pandagg.query module

```

class pandagg.query.Query(from_=None, mapping=None, identifier=None, client=None, index_name=None)
    Bases: pandagg.tree._tree.Tree

    Tree combination of query nodes.

    Mapping declaration is optional, but doing so validates query validity.

```

**add\_node** (*node*, *pid=None*)  
Add a new node object to the tree and make the parent as the root by default.  
The 'node' parameter refers to an instance of `Class::Node`.

**bind** (*client*, *index\_name=None*)

**bool** (*\*args*, *\*\*kwargs*)

**boost** (*\*args*, *\*\*kwargs*)

**constant\_score** (*\*args*, *\*\*kwargs*)

**classmethod deserialize** (*from\_*)

**dis\_max** (*\*args*, *\*\*kwargs*)

**execute** (*index=None*, *\*\*kwargs*)

**filter** (*\*args*, *\*\*kwargs*)

**function\_score** (*\*args*, *\*\*kwargs*)

**has\_child** (*\*args*, *\*\*kwargs*)

**has\_parent** (*\*args*, *\*\*kwargs*)

**must** (*\*args*, *\*\*kwargs*)

**must\_not** (*\*args*, *\*\*kwargs*)

**nested** (*\*args*, *\*\*kwargs*)

**node\_class**  
alias of `pandagg.node.query.abstract.QueryClause`

**parent\_id** (*\*args*, *\*\*kwargs*)

**pinned\_query** (*\*args*, *\*\*kwargs*)

**query** (*q*, *parent=None*, *child=None*, *parent\_param=None*, *child\_param=None*, *mode='add'*)  
Place query below a given parent.

**query\_dict** (*from\_=None*, *named=False*)  
Return None if no query clause.

**script\_score** (*\*args*, *\*\*kwargs*)

**set\_mapping** (*mapping*)

**should** (*\*args*, *\*\*kwargs*)

**class** `pandagg.query.Exists` (*field*, *\_name=None*)  
Bases: `pandagg.node.query.abstract.LeafQueryClause`  
**KEY** = 'exists'  
**tag**  
The readable node name for human. This attribute can be accessed and modified with `.` and `=` operator respectively.

**class** `pandagg.query.Fuzzy` (*field*, *\_name=None*, *\*\*body*)  
Bases: `pandagg.node.query.abstract.SingleFieldQueryClause`  
**KEY** = 'fuzzy'

**class** `pandagg.query.Ids` (*values*, *\_name=None*)  
Bases: `pandagg.node.query.abstract.LeafQueryClause`

```

    KEY = 'ids'

    serialize (named=False)

    tag
        The readable node name for human. This attribute can be accessed and modified with . and = operator
        respectively.

class pandagg.query.Prefix (field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'prefix'

class pandagg.query.Range (field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'range'

class pandagg.query.Regexp (field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'regexp'

class pandagg.query.Term (field, value, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'term'

    SHORT_TAG = 'value'

class pandagg.query.Terms (field, terms, _name=None, **body)
    Bases: pandagg.node.query.abstract.LeafQueryClause

    KEY = 'terms'

    classmethod deserialize (**body)

    tag
        The readable node name for human. This attribute can be accessed and modified with . and = operator
        respectively.

class pandagg.query.TermsSet (field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'terms_set'

class pandagg.query.Type (field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'type'

class pandagg.query.Wildcard (field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'wildcard'

class pandagg.query.Intervals (field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'intervals'

class pandagg.query.Match (field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'match'

    SHORT_TAG = 'query'

```

```
class pandagg.query.MatchBoolPrefix (field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'match_bool_prefix'

    SHORT_TAG = 'query'

class pandagg.query.MatchPhrase (field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'match_phrase'

    SHORT_TAG = 'query'

class pandagg.query.MatchPhrasePrefix (field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'match_phrase_prefix'

    SHORT_TAG = 'query'

class pandagg.query.MultiMatch (fields, _name=None, **body)
    Bases: pandagg.node.query.abstract.MultiFieldsQueryClause

    KEY = 'multi_match'

class pandagg.query.Common (field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'common'

class pandagg.query.QueryString (_name=None, **body)
    Bases: pandagg.node.query.abstract.LeafQueryClause

    KEY = 'query_string'

class pandagg.query.SimpleQueryString (_name=None, **body)
    Bases: pandagg.node.query.abstract.LeafQueryClause

    KEY = 'simple_string'

class pandagg.query.Bool (*args, **kwargs)
    Bases: pandagg.node.query.compound.CompoundClause

    DEFAULT_OPERATOR
        alias of pandagg.node.query._parameter_clause.Must

    KEY = 'bool'

    PARAMS_WHITELIST = ['should', 'must', 'must_not', 'filter', 'boost', 'minimum_should_m

class pandagg.query.Boosting (*args, **kwargs)
    Bases: pandagg.node.query.compound.CompoundClause

    DEFAULT_OPERATOR
        alias of pandagg.node.query._parameter_clause.Positive

    KEY = 'boosting'

    PARAMS_WHITELIST = ['positive', 'negative', 'negative_boost']

class pandagg.query.ConstantScore (*args, **kwargs)
    Bases: pandagg.node.query.compound.CompoundClause

    DEFAULT_OPERATOR
        alias of pandagg.node.query._parameter_clause.Filter
```

```

    KEY = 'constant_score'

    PARAMS_WHITELIST = ['filter', 'boost']

class pandagg.query.FunctionScore(*args, **kwargs)
    Bases: pandagg.node.query.compound.CompoundClause

    DEFAULT_OPERATOR
        alias of pandagg.node.query._parameter_clause.QueryP

    KEY = 'function_score'

    PARAMS_WHITELIST = ['query', 'boost', 'random_score', 'boost_mode', 'functions', 'max_

class pandagg.query.DisMax(*args, **kwargs)
    Bases: pandagg.node.query.compound.CompoundClause

    DEFAULT_OPERATOR
        alias of pandagg.node.query._parameter_clause.Queries

    KEY = 'dis_max'

    PARAMS_WHITELIST = ['queries', 'tie_breaker']

class pandagg.query.Nested(*args, **kwargs)
    Bases: pandagg.node.query.compound.CompoundClause

    DEFAULT_OPERATOR
        alias of pandagg.node.query._parameter_clause.QueryP

    KEY = 'nested'

    PARAMS_WHITELIST = ['path', 'query', 'score_mode', 'ignore_unmapped']

class pandagg.query.HasParent(*args, **kwargs)
    Bases: pandagg.node.query.compound.CompoundClause

    DEFAULT_OPERATOR
        alias of pandagg.node.query._parameter_clause.QueryP

    KEY = 'has_parent'

    PARAMS_WHITELIST = ['query', 'parent_type', 'score', 'ignore_unmapped']

class pandagg.query.HasChild(*args, **kwargs)
    Bases: pandagg.node.query.compound.CompoundClause

    DEFAULT_OPERATOR
        alias of pandagg.node.query._parameter_clause.QueryP

    KEY = 'has_child'

    PARAMS_WHITELIST = ['query', 'type', 'max_children', 'min_children', 'score_mode', 'ig

class pandagg.query.ParentId(*args, **kwargs)
    Bases: pandagg.node.query.compound.CompoundClause

    KEY = 'parent_id'

class pandagg.query.Shape(_name=None, **body)
    Bases: pandagg.node.query.abstract.LeafQueryClause

    KEY = 'shape'

class pandagg.query.GeoShape(field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

```

```
KEY = 'geo_shape'

class pandagg.query.GeoPolygone (field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'geo_polygon'

class pandagg.query.GeoDistance (field, location, distance, _name=None, **body)
    Bases: pandagg.node.query.abstract.LeafQueryClause

    KEY = 'geo_distance'

    classmethod deserialize (**body)

    tag
        The readable node name for human. This attribute can be accessed and modified with . and = operator
        respectively.

class pandagg.query.GeoBoundingBox (field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    KEY = 'geo_bounding_box'

class pandagg.query.DistanceFeature (field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    FLAT = True

    KEY = 'distance_feature'

class pandagg.query.MoreLikeThis (fields, _name=None, **body)
    Bases: pandagg.node.query.abstract.MultiFieldsQueryClause

    KEY = 'more_like_this'

class pandagg.query.Percolate (field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    FLAT = True

    KEY = 'percolate'

class pandagg.query.RankFeature (field, _name=None, **body)
    Bases: pandagg.node.query.abstract.SingleFieldQueryClause

    FLAT = True

    KEY = 'rank_feature'

class pandagg.query.Script (_name=None, **body)
    Bases: pandagg.node.query.abstract.LeafQueryClause

    KEY = 'script'

class pandagg.query.Wrapper (_name=None, **body)
    Bases: pandagg.node.query.abstract.LeafQueryClause

    KEY = 'wrapper'

class pandagg.query.ScriptScore (*args, **kwargs)
    Bases: pandagg.node.query.compound.CompoundClause

    DEFAULT_OPERATOR
        alias of pandagg.node.query._parameter_clause.QueryP

    KEY = 'script_score'
```



```
PARAMS_WHITELIST = ['query', 'script', 'min_score']

class pandagg.query.PinnedQuery(*args, **kwargs)
    Bases: pandagg.node.query.compound.CompoundClause

    DEFAULT_OPERATOR
        alias of pandagg.node.query._parameter_clause.Organic

    KEY = 'pinned'

    PARAMS_WHITELIST = ['ids', 'organic']
```

### 5.2.6 pandagg.utils module

```
class pandagg.utils.PrettyNode(pretty)
    Bases: object

    pretty

pandagg.utils.bool_if_required(conditions, operator='must')

pandagg.utils.equal_queries(d1, d2)
    Compares if two queries are equivalent (do not consider nested list orders).

pandagg.utils.ordered(obj)
```

## 5.3 Module contents



We want to make contributing to this project as easy and transparent as possible.

### 6.1 Our Development Process

We use github to host code, to track issues and feature requests, as well as accept pull requests.

### 6.2 Pull Requests

We actively welcome your pull requests.

1. Fork the repo and create your branch from `master`.
2. If you've added code that should be tested, add tests.
3. If you've changed APIs, update the documentation.
4. Ensure the test suite passes.
5. Make sure your code lints.

### 6.3 Any contributions you make will be under the MIT Software License

In short, when you submit code changes, your submissions are understood to be under the same [MIT License](#) that covers the project. Feel free to contact the maintainers if that's a concern.

## 6.4 Issues

We use GitHub issues to track public bugs. Please ensure your description is clear and has sufficient instructions to be able to reproduce the issue.

## 6.5 Report bugs using Github's issues

We use GitHub issues to track public bugs. Report a bug by [opening a new issue](#); it's that easy!

## 6.6 Write bug reports with detail, background, and sample code

**Great Bug Reports** tend to have:

- A quick summary and/or background
- Steps to reproduce
  - Be specific!
  - Give sample code if you can.
- What you expected would happen
- What actually happens
- Notes (possibly including why you think this might be happening, or stuff you tried that didn't work)

## 6.7 License

By contributing, you agree that your contributions will be licensed under its MIT License.

## 6.8 References

This document was adapted from the open-source contribution guidelines of [briandk's gist](#)

**pandagg** is a Python package providing a simple interface to manipulate Elasticsearch queries and aggregations. It brings the following features:

- flexible aggregation and search queries declaration
- query validation based on provided mapping
- parsing of aggregation results in handy format: interactive bucket tree, normalized tree or tabular breakdown
- mapping interactive navigation

## CHAPTER 7

---

### Installing

---

**pandagg** can be installed with **pip**:

```
$ pip install pandagg
```

Alternatively, you can grab the latest source code from [GitHub](#):

```
$ git clone git://github.com/alkemics/pandagg.git
$ python setup.py install
```



## CHAPTER 8

---

### Usage

---

The *User Guide* is the place to go to learn how to use the library and accomplish common tasks. The more in-depth *Advanced usage* guide is the place to go for deeply nested queries.

An example based on publicly available IMDB data is documented in repository *examples/imdb* directory, with a jupyter notebook to showcase some of *pandagg* functionalities: [here it is](#).

The *pandagg package* documentation provides API-level documentation.





## CHAPTER 9

---

### License

---

pandagg is made available under the MIT License. For more details, see [LICENSE.txt](#).



## CHAPTER 10

---

### Contributing

---

We happily welcome contributions, please see *Contributing to Pandagg* for details.



### p

- `pandagg`, 61
- `pandagg.agg`, 43
- `pandagg.client`, 49
- `pandagg.exceptions`, 49
- `pandagg.interactive`, 17
- `pandagg.interactive.abstract`, 15
- `pandagg.interactive.client`, 16
- `pandagg.interactive.index`, 16
- `pandagg.interactive.mapping`, 16
- `pandagg.interactive.response`, 16
- `pandagg.mapping`, 50
- `pandagg.node`, 38
- `pandagg.node.agg`, 25
- `pandagg.node.agg.abstract`, 17
- `pandagg.node.agg.bucket`, 19
- `pandagg.node.agg.deserializer`, 22
- `pandagg.node.agg.metric`, 22
- `pandagg.node.agg.pipeline`, 23
- `pandagg.node.mapping`, 31
- `pandagg.node.mapping.abstract`, 25
- `pandagg.node.mapping.deserializer`, 26
- `pandagg.node.mapping.field_datatypes`, 26
- `pandagg.node.mapping.meta_fields`, 30
- `pandagg.node.mixins`, 38
- `pandagg.node.query`, 37
- `pandagg.node.query.abstract`, 31
- `pandagg.node.query.compound`, 32
- `pandagg.node.query.deserializer`, 33
- `pandagg.node.query.full_text`, 33
- `pandagg.node.query.geo`, 34
- `pandagg.node.query.joining`, 34
- `pandagg.node.query.shape`, 35
- `pandagg.node.query.span`, 35
- `pandagg.node.query.specialized`, 35
- `pandagg.node.query.specialized_compound`, 36
- `pandagg.node.query.term_level`, 36
- `pandagg.node.response`, 38
- `pandagg.node.response.bucket`, 37
- `pandagg.node.types`, 38
- `pandagg.query`, 55
- `pandagg.tree`, 43
- `pandagg.tree.agg`, 38
- `pandagg.tree.mapping`, 40
- `pandagg.tree.query`, 41
- `pandagg.tree.response`, 42
- `pandagg.utils`, 61



## A

AbsentMappingFieldError, 49  
 add\_node() (*pandagg.agg.Aggreg method*), 43  
 add\_node() (*pandagg.query.Query method*), 55  
 add\_node() (*pandagg.tree.agg.Aggreg method*), 38  
 add\_node() (*pandagg.tree.query.Query method*), 41  
 Agg (*class in pandagg.agg*), 43  
 Agg (*class in pandagg.tree.agg*), 38  
 agg() (*pandagg.agg.Aggreg method*), 43  
 agg() (*pandagg.interactive.index.Index method*), 16  
 agg() (*pandagg.tree.agg.Aggreg method*), 38  
 AggNode (*class in pandagg.node.agg.abstract*), 17  
 Alias (*class in pandagg.mapping*), 54  
 Alias (*class in pandagg.node.mapping.field\_datatypes*), 26  
 Aliases (*class in pandagg.interactive.index*), 16  
 applied\_nested\_path\_at\_node() (*pandagg.agg.Aggreg method*), 43  
 applied\_nested\_path\_at\_node() (*pandagg.tree.agg.Aggreg method*), 39  
 as\_mapping() (*in pandagg.interactive.mapping*), 16  
 attr\_name (*pandagg.node.response.bucket.Bucket attribute*), 37  
 Avg (*class in pandagg.agg*), 46  
 Avg (*class in pandagg.node.agg.metric*), 22  
 AvgBucket (*class in pandagg.agg*), 48  
 AvgBucket (*class in pandagg.node.agg.pipeline*), 23

## B

Binary (*class in pandagg.mapping*), 52  
 Binary (*class in pandagg.node.mapping.field\_datatypes*), 26  
 bind() (*pandagg.agg.Aggreg method*), 43  
 bind() (*pandagg.query.Query method*), 56  
 bind() (*pandagg.tree.agg.Aggreg method*), 39  
 bind() (*pandagg.tree.query.Query method*), 41  
 BLACKLISTED\_MAPPING\_TYPES (*pandagg.agg.Terms attribute*), 44

BLACKLISTED\_MAPPING\_TYPES (*pandagg.agg.ValueCount attribute*), 47  
 BLACKLISTED\_MAPPING\_TYPES (*pandagg.node.agg.abstract.AggNode attribute*), 17  
 BLACKLISTED\_MAPPING\_TYPES (*pandagg.node.agg.bucket.Missing attribute*), 20  
 BLACKLISTED\_MAPPING\_TYPES (*pandagg.node.agg.bucket.Terms attribute*), 21  
 BLACKLISTED\_MAPPING\_TYPES (*pandagg.node.agg.metric.ValueCount attribute*), 23  
 BLACKLISTED\_MAPPING\_TYPES (*pandagg.node.mixins.FieldValidityMixin attribute*), 38  
 body() (*pandagg.node.mapping.abstract.Field method*), 25  
 Bool (*class in pandagg.node.query.compound*), 32  
 Bool (*class in pandagg.query*), 58  
 bool() (*pandagg.query.Query method*), 56  
 bool() (*pandagg.tree.query.Query method*), 41  
 bool\_if\_required() (*in module pandagg.utils*), 61  
 Boolean (*class in pandagg.mapping*), 52  
 Boolean (*class in pandagg.node.mapping.field\_datatypes*), 26  
 boost() (*pandagg.query.Query method*), 56  
 boost() (*pandagg.tree.query.Query method*), 41  
 Boosting (*class in pandagg.node.query.compound*), 32  
 Boosting (*class in pandagg.query*), 58  
 Bucket (*class in pandagg.node.response.bucket*), 37  
 bucket\_properties() (*pandagg.tree.response.ResponseTree method*), 42  
 BucketAggNode (*class in pandagg.node.agg.abstract*), 18  
 BucketScript (*class in pandagg.agg*), 49  
 BucketScript (*class in pandagg.node.agg.pipeline*), 23  
 BucketSelector (*class in pandagg.agg*), 49

BucketSelector (class in pandagg.node.agg.pipeline), 24  
 BucketSort (class in pandagg.agg), 49  
 BucketSort (class in pandagg.node.agg.pipeline), 24  
 Byte (class in pandagg.mapping), 51  
 Byte (class in pandagg.node.mapping.field\_datatypes), 26

## C

Cardinality (class in pandagg.agg), 46  
 Cardinality (class in pandagg.node.agg.metric), 22  
 Common (class in pandagg.node.query.full\_text), 33  
 Common (class in pandagg.query), 58  
 Completion (class in pandagg.mapping), 53  
 Completion (class in pandagg.node.mapping.field\_datatypes), 26  
 CompoundClause (class in pandagg.node.query.compound), 32  
 constant\_score() (pandagg.query.Query method), 56  
 constant\_score() (pandagg.tree.query.Query method), 41  
 ConstantScore (class in pandagg.node.query.compound), 32  
 ConstantScore (class in pandagg.query), 58  
 contains() (pandagg.mapping.Mapping method), 50  
 contains() (pandagg.tree.mapping.Mapping method), 40  
 CumulativeSum (class in pandagg.agg), 49  
 CumulativeSum (class in pandagg.node.agg.pipeline), 24

## D

Date (class in pandagg.mapping), 52  
 Date (class in pandagg.node.mapping.field\_datatypes), 26  
 DateHistogram (class in pandagg.agg), 45  
 DateHistogram (class in pandagg.node.agg.bucket), 19  
 DateNanos (class in pandagg.mapping), 52  
 DateNanos (class in pandagg.node.mapping.field\_datatypes), 26  
 DateRange (class in pandagg.mapping), 52  
 DateRange (class in pandagg.node.agg.bucket), 19  
 DateRange (class in pandagg.node.mapping.field\_datatypes), 26  
 deepest\_linear\_bucket\_agg (pandagg.agg.Aggregate attribute), 43  
 deepest\_linear\_bucket\_agg (pandagg.tree.agg.Aggregate attribute), 39

in DEFAULT\_OPERATOR (pandagg.node.query.compound.Bool attribute), 32  
 DEFAULT\_OPERATOR (pandagg.node.query.compound.Boosting attribute), 32  
 DEFAULT\_OPERATOR (pandagg.node.query.compound.CompoundClause attribute), 32  
 DEFAULT\_OPERATOR (pandagg.node.query.compound.ConstantScore attribute), 32  
 DEFAULT\_OPERATOR (pandagg.node.query.compound.DisMax attribute), 33  
 DEFAULT\_OPERATOR (pandagg.node.query.compound.FunctionScore attribute), 33  
 DEFAULT\_OPERATOR (pandagg.node.query.joining.HasChild attribute), 34  
 in DEFAULT\_OPERATOR (pandagg.node.query.joining.HasParent attribute), 35  
 DEFAULT\_OPERATOR (pandagg.node.query.joining.Nested attribute), 35  
 in DEFAULT\_OPERATOR (pandagg.node.query.specialized\_compound.Pinned attribute), 36  
 DEFAULT\_OPERATOR (pandagg.node.query.specialized\_compound.Script attribute), 36  
 DEFAULT\_OPERATOR (pandagg.query.Bool attribute), 58  
 DEFAULT\_OPERATOR (pandagg.query.Boosting attribute), 58  
 DEFAULT\_OPERATOR (pandagg.query.ConstantScore attribute), 58  
 DEFAULT\_OPERATOR (pandagg.query.DisMax attribute), 59  
 DEFAULT\_OPERATOR (pandagg.query.FunctionScore attribute), 59  
 DEFAULT\_OPERATOR (pandagg.query.HasChild attribute), 59  
 DEFAULT\_OPERATOR (pandagg.query.HasParent attribute), 59  
 DEFAULT\_OPERATOR (pandagg.query.Nested attribute), 59  
 DEFAULT\_OPERATOR (pandagg.query.PinnedQuery attribute), 61  
 DEFAULT\_OPERATOR (pandagg.query.ScriptScore attribute), 60  
 DEFAULT\_OTHER\_KEY (pandagg.agg.Filters attribute), 44  
 DEFAULT\_OTHER\_KEY (pandagg.node.agg.bucket.Filters attribute), 20  
 DEFAULT\_OUTPUT (pandagg.agg.Aggregate attribute), 43  
 DEFAULT\_OUTPUT (pandagg.tree.agg.Aggregate attribute), 38  
 DenseVector (class in pandagg.mapping), 54  
 DenseVector (class in pandagg.node.mapping.field\_datatypes), 26  
 in Derivative (class in pandagg.agg), 48



- Derivative (class in *pandagg.node.agg.pipeline*), 24
- deserialize() (*pandagg.agg.Agg* class method), 43
- deserialize() (*pandagg.mapping.Mapping* class method), 50
- deserialize() (*pandagg.node.agg.abstract.AggNode* class method), 17
- deserialize() (*pandagg.node.mapping.abstract.Field* class method), 25
- deserialize() (*pandagg.node.query.abstract.QueryClause* class method), 31
- deserialize() (*pandagg.node.query.abstract.SingleFieldQueryClause* class method), 32
- deserialize() (*pandagg.node.query.compound.CompoundClause* class method), 32
- deserialize() (*pandagg.node.query.geo.GeoDistance* class method), 34
- deserialize() (*pandagg.node.query.term\_level.Terms* class method), 37
- deserialize() (*pandagg.query.GeoDistance* class method), 60
- deserialize() (*pandagg.query.Query* class method), 56
- deserialize() (*pandagg.query.Terms* class method), 57
- deserialize() (*pandagg.tree.agg.Agg* class method), 39
- deserialize() (*pandagg.tree.mapping.Mapping* class method), 40
- deserialize() (*pandagg.tree.query.Query* class method), 41
- deserialize\_agg() (in module *pandagg.node.agg.deserializer*), 22
- deserialize\_field() (in module *pandagg.node.mapping.deserializer*), 26
- deserialize\_node() (in module *pandagg.node.query.deserializer*), 33
- dis\_max() (*pandagg.query.Query* method), 56
- dis\_max() (*pandagg.tree.query.Query* method), 41
- DisMax (class in *pandagg.node.query.compound*), 33
- DisMax (class in *pandagg.query*), 59
- display\_name (*pandagg.node.response.bucket.Bucket* attribute), 38
- display\_name\_with\_value (*pandagg.node.response.bucket.Bucket* attribute), 38
- DISPLAY\_PATTERN (*pandagg.mapping.Nested* attribute), 52
- DISPLAY\_PATTERN (*pandagg.mapping.Object* attribute), 52
- DISPLAY\_PATTERN (*pandagg.node.mapping.abstract.Field* attribute), 25
- DISPLAY\_PATTERN (*pandagg.node.mapping.field\_datatypes.Nested* attribute), 29
- DISPLAY\_PATTERN (*pandagg.node.mapping.field\_datatypes.Object* attribute), 29
- DistanceFeature (class in *pandagg.node.query.specialized*), 35
- DistanceFeature (class in *pandagg.query*), 60
- Double (class in *pandagg.mapping*), 51
- Double (class in *pandagg.node.mapping.field\_datatypes*), 27
- DoubleRange (class in *pandagg.mapping*), 52
- DoubleRange (class in *pandagg.node.mapping.field\_datatypes*), 27
- Elasticsearch (class in *pandagg.client*), 49
- Elasticsearch (class in *pandagg.interactive.client*), 16
- equal\_queries() (in module *pandagg.utils*), 61
- execute() (*pandagg.agg.Agg* method), 43
- execute() (*pandagg.query.Query* method), 56
- execute() (*pandagg.tree.agg.Agg* method), 39
- execute() (*pandagg.tree.query.Query* method), 41
- Exists (class in *pandagg.node.query.term\_level*), 36
- Exists (class in *pandagg.query*), 56
- ExtendedStats (class in *pandagg.agg*), 47
- ExtendedStats (class in *pandagg.node.agg.metric*), 22
- ExtendedStatsBucket (class in *pandagg.agg*), 48
- ExtendedStatsBucket (class in *pandagg.node.agg.pipeline*), 24
- extract\_bucket\_value() (*pandagg.node.agg.abstract.AggNode* class method), 17
- extract\_buckets() (*pandagg.node.agg.abstract.AggNode* method), 17
- extract\_buckets() (*pandagg.node.agg.abstract.BucketAggNode* method), 18
- extract\_buckets() (*pandagg.node.agg.abstract.MetricAgg* method), 18
- extract\_buckets() (*pandagg.node.agg.abstract.MultipleBucketAgg* method), 18
- extract\_buckets() (*pandagg.node.agg.abstract.UniqueBucketAgg* method), 19
- FetchIndices() (*pandagg.client.Elasticsearch* method), 49
- FetchIndices() (*pandagg.interactive.client.Elasticsearch* method), 16
- Field (class in *pandagg.node.mapping.abstract*), 25

- FieldNames (class in pandagg.mapping), 55
- FieldNames (class pandagg.node.mapping.meta\_fields), 30
- FieldOrScriptMetricAgg (class pandagg.node.agg.abstract), 18
- FieldValidityMixin (class pandagg.node.mixins), 38
- Filter (class in pandagg.agg), 45
- Filter (class in pandagg.node.agg.bucket), 20
- filter() (pandagg.query.Query method), 56
- filter() (pandagg.tree.query.Query method), 41
- Filters (class in pandagg.agg), 44
- Filters (class in pandagg.node.agg.bucket), 20
- FLAT (pandagg.node.query.abstract.SingleFieldQueryClause attribute), 32
- FLAT (pandagg.node.query.specialized.DistanceFeature attribute), 35
- FLAT (pandagg.node.query.specialized.Percolate attribute), 35
- FLAT (pandagg.node.query.specialized.RankFeature attribute), 35
- FLAT (pandagg.query.DistanceFeature attribute), 60
- FLAT (pandagg.query.Percolate attribute), 60
- FLAT (pandagg.query.RankFeature attribute), 60
- Flattened (class in pandagg.mapping), 54
- Flattened (class pandagg.node.mapping.field\_datatypes), 27
- Float (class in pandagg.mapping), 52
- Float (class in pandagg.node.mapping.field\_datatypes), 27
- FloatRange (class in pandagg.mapping), 52
- FloatRange (class pandagg.node.mapping.field\_datatypes), 27
- from\_key (pandagg.agg.Range attribute), 45
- from\_key (pandagg.node.agg.bucket.Range attribute), 21
- function\_score() (pandagg.query.Query method), 56
- function\_score() (pandagg.tree.query.Query method), 41
- FunctionScore (class pandagg.node.query.compound), 33
- FunctionScore (class in pandagg.query), 59
- Fuzzy (class in pandagg.node.query.term\_level), 36
- Fuzzy (class in pandagg.query), 56
- GeoBound (class in pandagg.agg), 47
- GeoBound (class in pandagg.node.agg.metric), 22
- GeoBoundingBox (class in pandagg.node.query.geo), 34
- GeoBoundingBox (class in pandagg.query), 60
- GeoCentroid (class in pandagg.agg), 47
- in GeoCentroid (class in pandagg.node.agg.metric), 22
- GeoDistance (class in pandagg.node.query.geo), 34
- in GeoDistance (class in pandagg.query), 60
- GeoPoint (class in pandagg.mapping), 53
- in GeoPoint (class in pandagg.node.mapping.field\_datatypes), 27
- GeoPolygone (class in pandagg.node.query.geo), 34
- GeoPolygone (class in pandagg.query), 60
- GeoShape (class in pandagg.mapping), 53
- GeoShape (class in pandagg.node.mapping.field\_datatypes), 27
- GeoShape (class in pandagg.node.query.geo), 34
- GeoShape (class in pandagg.query), 59
- get\_bucket\_filter() (pandagg.interactive.response.IResponse method), 16
- get\_bucket\_filter() (pandagg.tree.response.ResponseTree method), 42
- get\_filter() (pandagg.agg.DateHistogram method), 45
- get\_filter() (pandagg.agg.Filter method), 45
- get\_filter() (pandagg.agg.Filters method), 45
- get\_filter() (pandagg.agg.Global method), 45
- get\_filter() (pandagg.agg.Histogram method), 45
- get\_filter() (pandagg.agg.Nested method), 46
- get\_filter() (pandagg.agg.Range method), 45
- get\_filter() (pandagg.agg.ReverseNested method), 46
- get\_filter() (pandagg.agg.Terms method), 44
- get\_filter() (pandagg.node.agg.abstract.AggregateNode method), 17
- get\_filter() (pandagg.node.agg.abstract.BucketAggregateNode method), 18
- get\_filter() (pandagg.node.agg.abstract.MetricAggregate method), 18
- get\_filter() (pandagg.node.agg.abstract.MultipleBucketAggregate method), 18
- get\_filter() (pandagg.node.agg.abstract.Pipeline method), 19
- get\_filter() (pandagg.node.agg.abstract.UniqueBucketAggregate method), 19
- get\_filter() (pandagg.node.agg.bucket.DateHistogram method), 19
- get\_filter() (pandagg.node.agg.bucket.Filter method), 20
- get\_filter() (pandagg.node.agg.bucket.Filters method), 20
- get\_filter() (pandagg.node.agg.bucket.Global method), 20
- get\_filter() (pandagg.node.agg.bucket.Histogram method), 20
- get\_filter() (pandagg.node.agg.bucket.Missing

- method*), 21
  - `get_filter()` (*pandagg.node.agg.bucket.Nested method*), 21
  - `get_filter()` (*pandagg.node.agg.bucket.Range method*), 21
  - `get_filter()` (*pandagg.node.agg.bucket.ReverseNested method*), 21
  - `get_filter()` (*pandagg.node.agg.bucket.Terms method*), 21
  - `Global` (*class in pandagg.agg*), 45
  - `Global` (*class in pandagg.node.agg.bucket*), 20
  - `groupby()` (*pandagg.agg.Agg method*), 43
  - `groupby()` (*pandagg.interactive.index.Index method*), 16
  - `groupby()` (*pandagg.tree.agg.Agg method*), 39
- ## H
- `HalfFloat` (*class in pandagg.mapping*), 51
  - `HalfFloat` (*class in pandagg.node.mapping.field\_datatypes*), 27
  - `has_child()` (*pandagg.query.Query method*), 56
  - `has_child()` (*pandagg.tree.query.Query method*), 41
  - `has_parent()` (*pandagg.query.Query method*), 56
  - `has_parent()` (*pandagg.tree.query.Query method*), 41
  - `HasChild` (*class in pandagg.node.query.joining*), 34
  - `HasChild` (*class in pandagg.query*), 59
  - `HasParent` (*class in pandagg.node.query.joining*), 35
  - `HasParent` (*class in pandagg.query*), 59
  - `Histogram` (*class in pandagg.agg*), 45
  - `Histogram` (*class in pandagg.mapping*), 54
  - `Histogram` (*class in pandagg.node.agg.bucket*), 20
  - `Histogram` (*class in pandagg.node.mapping.field\_datatypes*), 27
- ## I
- `Id` (*class in pandagg.mapping*), 55
  - `Id` (*class in pandagg.node.mapping.meta\_fields*), 30
  - `Ids` (*class in pandagg.node.query.term\_level*), 36
  - `Ids` (*class in pandagg.query*), 56
  - `Ignored` (*class in pandagg.mapping*), 55
  - `Ignored` (*class in pandagg.node.mapping.meta\_fields*), 30
  - `IMapping` (*class in pandagg.interactive.mapping*), 16
  - `IMapping` (*class in pandagg.mapping*), 51
  - `IMPLICIT_KEYED` (*pandagg.agg.Filters attribute*), 44
  - `IMPLICIT_KEYED` (*pandagg.node.agg.abstract.MultipleBucketAgg attribute*), 18
  - `IMPLICIT_KEYED` (*pandagg.node.agg.bucket.Filters attribute*), 20
  - `Index` (*class in pandagg.interactive.index*), 16
  - `Index` (*class in pandagg.mapping*), 54
  - `Index` (*class in pandagg.node.mapping.meta\_fields*), 30
  - `Indices` (*class in pandagg.interactive.index*), 16
  - `Integer` (*class in pandagg.mapping*), 51
  - `Integer` (*class in pandagg.node.mapping.field\_datatypes*), 28
  - `IntegerRange` (*class in pandagg.mapping*), 52
  - `IntegerRange` (*class in pandagg.node.mapping.field\_datatypes*), 28
  - `Intervals` (*class in pandagg.node.query.full\_text*), 33
  - `Intervals` (*class in pandagg.query*), 57
  - `InvalidAggregation`, 50
  - `InvalidOperationMappingFieldError`, 50
  - `IP` (*class in pandagg.mapping*), 53
  - `IP` (*class in pandagg.node.mapping.field\_datatypes*), 27
  - `IResponse` (*class in pandagg.interactive.response*), 16
  - `is_valid_attr_name()` (*in pandagg.interactive.abstract*), 15
- ## J
- `Join` (*class in pandagg.mapping*), 53
  - `Join` (*class in pandagg.node.mapping.field\_datatypes*), 28
- ## K
- `KEY` (*pandagg.agg.Avg attribute*), 46
  - `KEY` (*pandagg.agg.AvgBucket attribute*), 48
  - `KEY` (*pandagg.agg.BucketScript attribute*), 49
  - `KEY` (*pandagg.agg.BucketSelector attribute*), 49
  - `KEY` (*pandagg.agg.BucketSort attribute*), 49
  - `KEY` (*pandagg.agg.Cardinality attribute*), 46
  - `KEY` (*pandagg.agg.CumulativeSum attribute*), 49
  - `KEY` (*pandagg.agg.DateHistogram attribute*), 45
  - `KEY` (*pandagg.agg.Derivative attribute*), 48
  - `KEY` (*pandagg.agg.ExtendedStats attribute*), 47
  - `KEY` (*pandagg.agg.ExtendedStatsBucket attribute*), 48
  - `KEY` (*pandagg.agg.Filter attribute*), 45
  - `KEY` (*pandagg.agg.Filters attribute*), 44
  - `KEY` (*pandagg.agg.GeoBound attribute*), 47
  - `KEY` (*pandagg.agg.GeoCentroid attribute*), 47
  - `KEY` (*pandagg.agg.Global attribute*), 45
  - `KEY` (*pandagg.agg.Histogram attribute*), 45
  - `KEY` (*pandagg.agg.Max attribute*), 46
  - `KEY` (*pandagg.agg.MaxBucket attribute*), 48
  - `KEY` (*pandagg.agg.Min attribute*), 46
  - `KEY` (*pandagg.agg.MinBucket attribute*), 48
  - `KEY` (*pandagg.agg.MovingAvg attribute*), 49
  - `KEY` (*pandagg.agg.Nested attribute*), 46
  - `KEY` (*pandagg.agg.PercentileRanks attribute*), 47
  - `KEY` (*pandagg.agg.Percentiles attribute*), 47
  - `KEY` (*pandagg.agg.PercentilesBucket attribute*), 48
  - `KEY` (*pandagg.agg.Range attribute*), 45
  - `KEY` (*pandagg.agg.ReverseNested attribute*), 46
  - `KEY` (*pandagg.agg.SerialDiff attribute*), 49

- KEY (*pandagg.agg.Stats* attribute), 47
- KEY (*pandagg.agg.StatsBucket* attribute), 48
- KEY (*pandagg.agg.Sum* attribute), 46
- KEY (*pandagg.agg.SumBucket* attribute), 48
- KEY (*pandagg.agg.Terms* attribute), 44
- KEY (*pandagg.agg.TopHits* attribute), 47
- KEY (*pandagg.agg.ValueCount* attribute), 47
- KEY (*pandagg.mapping.Alias* attribute), 54
- KEY (*pandagg.mapping.Binary* attribute), 52
- KEY (*pandagg.mapping.Boolean* attribute), 52
- KEY (*pandagg.mapping.Byte* attribute), 51
- KEY (*pandagg.mapping.Completion* attribute), 53
- KEY (*pandagg.mapping.Date* attribute), 52
- KEY (*pandagg.mapping.DateNanos* attribute), 52
- KEY (*pandagg.mapping.DateRange* attribute), 52
- KEY (*pandagg.mapping.DenseVector* attribute), 54
- KEY (*pandagg.mapping.Double* attribute), 51
- KEY (*pandagg.mapping.DoubleRange* attribute), 52
- KEY (*pandagg.mapping.FieldNames* attribute), 55
- KEY (*pandagg.mapping.Flattened* attribute), 54
- KEY (*pandagg.mapping.Float* attribute), 52
- KEY (*pandagg.mapping.FloatRange* attribute), 52
- KEY (*pandagg.mapping.GeoPoint* attribute), 53
- KEY (*pandagg.mapping.GeoShape* attribute), 53
- KEY (*pandagg.mapping.HalfFloat* attribute), 51
- KEY (*pandagg.mapping.Histogram* attribute), 54
- KEY (*pandagg.mapping.Id* attribute), 55
- KEY (*pandagg.mapping.Ignored* attribute), 55
- KEY (*pandagg.mapping.Index* attribute), 54
- KEY (*pandagg.mapping.Integer* attribute), 51
- KEY (*pandagg.mapping.IntegerRange* attribute), 52
- KEY (*pandagg.mapping.IP* attribute), 53
- KEY (*pandagg.mapping.Join* attribute), 53
- KEY (*pandagg.mapping.Keyword* attribute), 51
- KEY (*pandagg.mapping.Long* attribute), 51
- KEY (*pandagg.mapping.LongRange* attribute), 52
- KEY (*pandagg.mapping.MapperAnnotatedText* attribute), 53
- KEY (*pandagg.mapping.MapperMurMur3* attribute), 53
- KEY (*pandagg.mapping.Meta* attribute), 55
- KEY (*pandagg.mapping.Nested* attribute), 52
- KEY (*pandagg.mapping.Object* attribute), 52
- KEY (*pandagg.mapping.Percolator* attribute), 53
- KEY (*pandagg.mapping.RankFeature* attribute), 54
- KEY (*pandagg.mapping.RankFeatures* attribute), 54
- KEY (*pandagg.mapping.Routing* attribute), 55
- KEY (*pandagg.mapping.ScaledFloat* attribute), 52
- KEY (*pandagg.mapping.SearchAsYouType* attribute), 54
- KEY (*pandagg.mapping.Shape* attribute), 54
- KEY (*pandagg.mapping.Short* attribute), 51
- KEY (*pandagg.mapping.Size* attribute), 55
- KEY (*pandagg.mapping.Source* attribute), 55
- KEY (*pandagg.mapping.SparseVector* attribute), 54
- KEY (*pandagg.mapping.Text* attribute), 51
- KEY (*pandagg.mapping.TokenCount* attribute), 53
- KEY (*pandagg.mapping.Type* attribute), 55
- KEY (*pandagg.node.agg.abstract.AggNode* attribute), 17
- KEY (*pandagg.node.agg.abstract.ScriptPipeline* attribute), 19
- KEY (*pandagg.node.agg.abstract.ShadowRoot* attribute), 19
- KEY (*pandagg.node.agg.bucket.DateHistogram* attribute), 19
- KEY (*pandagg.node.agg.bucket.DateRange* attribute), 20
- KEY (*pandagg.node.agg.bucket.Filter* attribute), 20
- KEY (*pandagg.node.agg.bucket.Filters* attribute), 20
- KEY (*pandagg.node.agg.bucket.Global* attribute), 20
- KEY (*pandagg.node.agg.bucket.Histogram* attribute), 20
- KEY (*pandagg.node.agg.bucket.Missing* attribute), 21
- KEY (*pandagg.node.agg.bucket.Nested* attribute), 21
- KEY (*pandagg.node.agg.bucket.Range* attribute), 21
- KEY (*pandagg.node.agg.bucket.ReverseNested* attribute), 21
- KEY (*pandagg.node.agg.bucket.Terms* attribute), 21
- KEY (*pandagg.node.agg.metric.Avg* attribute), 22
- KEY (*pandagg.node.agg.metric.Cardinality* attribute), 22
- KEY (*pandagg.node.agg.metric.ExtendedStats* attribute), 22
- KEY (*pandagg.node.agg.metric.GeoBound* attribute), 22
- KEY (*pandagg.node.agg.metric.GeoCentroid* attribute), 22
- KEY (*pandagg.node.agg.metric.Max* attribute), 22
- KEY (*pandagg.node.agg.metric.Min* attribute), 22
- KEY (*pandagg.node.agg.metric.PercentileRanks* attribute), 23
- KEY (*pandagg.node.agg.metric.Percentiles* attribute), 23
- KEY (*pandagg.node.agg.metric.Stats* attribute), 23
- KEY (*pandagg.node.agg.metric.Sum* attribute), 23
- KEY (*pandagg.node.agg.metric.TopHits* attribute), 23
- KEY (*pandagg.node.agg.metric.ValueCount* attribute), 23
- KEY (*pandagg.node.agg.pipeline.AvgBucket* attribute), 23
- KEY (*pandagg.node.agg.pipeline.BucketScript* attribute), 24
- KEY (*pandagg.node.agg.pipeline.BucketSelector* attribute), 24
- KEY (*pandagg.node.agg.pipeline.BucketSort* attribute), 24
- KEY (*pandagg.node.agg.pipeline.CumulativeSum* attribute), 24
- KEY (*pandagg.node.agg.pipeline.Derivative* attribute), 24
- KEY (*pandagg.node.agg.pipeline.ExtendedStatsBucket* attribute), 24
- KEY (*pandagg.node.agg.pipeline.MaxBucket* attribute), 24
- KEY (*pandagg.node.agg.pipeline.MinBucket* attribute), 24
- KEY (*pandagg.node.agg.pipeline.MovingAvg* attribute),



- 25
- KEY (*pandagg.node.agg.pipeline.PercentilesBucket attribute*), 25
- KEY (*pandagg.node.agg.pipeline.SerialDiff attribute*), 25
- KEY (*pandagg.node.agg.pipeline.StatsBucket attribute*), 25
- KEY (*pandagg.node.agg.pipeline.SumBucket attribute*), 25
- KEY (*pandagg.node.mapping.abstract.Field attribute*), 25
- KEY (*pandagg.node.mapping.field\_datatypes.Alias attribute*), 26
- KEY (*pandagg.node.mapping.field\_datatypes.Binary attribute*), 26
- KEY (*pandagg.node.mapping.field\_datatypes.Boolean attribute*), 26
- KEY (*pandagg.node.mapping.field\_datatypes.Byte attribute*), 26
- KEY (*pandagg.node.mapping.field\_datatypes.Completion attribute*), 26
- KEY (*pandagg.node.mapping.field\_datatypes.Date attribute*), 26
- KEY (*pandagg.node.mapping.field\_datatypes.DateNanos attribute*), 26
- KEY (*pandagg.node.mapping.field\_datatypes.DateRange attribute*), 26
- KEY (*pandagg.node.mapping.field\_datatypes.DenseVector attribute*), 27
- KEY (*pandagg.node.mapping.field\_datatypes.Double attribute*), 27
- KEY (*pandagg.node.mapping.field\_datatypes.DoubleRange attribute*), 27
- KEY (*pandagg.node.mapping.field\_datatypes.Flattened attribute*), 27
- KEY (*pandagg.node.mapping.field\_datatypes.Float attribute*), 27
- KEY (*pandagg.node.mapping.field\_datatypes.FloatRange attribute*), 27
- KEY (*pandagg.node.mapping.field\_datatypes.GeoPoint attribute*), 27
- KEY (*pandagg.node.mapping.field\_datatypes.GeoShape attribute*), 27
- KEY (*pandagg.node.mapping.field\_datatypes.HalfFloat attribute*), 27
- KEY (*pandagg.node.mapping.field\_datatypes.Histogram attribute*), 27
- KEY (*pandagg.node.mapping.field\_datatypes.Integer attribute*), 28
- KEY (*pandagg.node.mapping.field\_datatypes.IntegerRange attribute*), 28
- KEY (*pandagg.node.mapping.field\_datatypes.IP attribute*), 28
- KEY (*pandagg.node.mapping.field\_datatypes.Join attribute*), 28
- KEY (*pandagg.node.mapping.field\_datatypes.Keyword attribute*), 28
- KEY (*pandagg.node.mapping.field\_datatypes.Long attribute*), 28
- KEY (*pandagg.node.mapping.field\_datatypes.LongRange attribute*), 28
- KEY (*pandagg.node.mapping.field\_datatypes.MapperAnnotatedText attribute*), 28
- KEY (*pandagg.node.mapping.field\_datatypes.MapperMurMur3 attribute*), 28
- KEY (*pandagg.node.mapping.field\_datatypes.Nested attribute*), 29
- KEY (*pandagg.node.mapping.field\_datatypes.Object attribute*), 29
- KEY (*pandagg.node.mapping.field\_datatypes.Percolator attribute*), 29
- KEY (*pandagg.node.mapping.field\_datatypes.RankFeature attribute*), 29
- KEY (*pandagg.node.mapping.field\_datatypes.RankFeatures attribute*), 29
- KEY (*pandagg.node.mapping.field\_datatypes.ScaledFloat attribute*), 29
- KEY (*pandagg.node.mapping.field\_datatypes.SearchAsYouType attribute*), 29
- KEY (*pandagg.node.mapping.field\_datatypes.Shape attribute*), 29
- KEY (*pandagg.node.mapping.field\_datatypes.Short attribute*), 29
- KEY (*pandagg.node.mapping.field\_datatypes.SparseVector attribute*), 30
- KEY (*pandagg.node.mapping.field\_datatypes.Text attribute*), 30
- KEY (*pandagg.node.mapping.field\_datatypes.TokenCount attribute*), 30
- KEY (*pandagg.node.mapping.meta\_fields.FieldNames attribute*), 30
- KEY (*pandagg.node.mapping.meta\_fields.Id attribute*), 30
- KEY (*pandagg.node.mapping.meta\_fields.Ignored attribute*), 30
- KEY (*pandagg.node.mapping.meta\_fields.Index attribute*), 30
- KEY (*pandagg.node.mapping.meta\_fields.Meta attribute*), 30
- KEY (*pandagg.node.mapping.meta\_fields.Routing attribute*), 31
- KEY (*pandagg.node.mapping.meta\_fields.Size attribute*), 31
- KEY (*pandagg.node.mapping.meta\_fields.Source attribute*), 31
- KEY (*pandagg.node.mapping.meta\_fields.Type attribute*), 31
- KEY (*pandagg.node.query.abstract.QueryClause attribute*), 31
- KEY (*pandagg.node.query.compound.Bool attribute*), 32

- KEY (*pandagg.node.query.compound.Boosting* attribute), 32
- KEY (*pandagg.node.query.compound.ConstantScore* attribute), 33
- KEY (*pandagg.node.query.compound.DisMax* attribute), 33
- KEY (*pandagg.node.query.compound.FunctionScore* attribute), 33
- KEY (*pandagg.node.query.full\_text.Common* attribute), 33
- KEY (*pandagg.node.query.full\_text.Intervals* attribute), 33
- KEY (*pandagg.node.query.full\_text.Match* attribute), 33
- KEY (*pandagg.node.query.full\_text.MatchBoolPrefix* attribute), 33
- KEY (*pandagg.node.query.full\_text.MatchPhrase* attribute), 33
- KEY (*pandagg.node.query.full\_text.MatchPhrasePrefix* attribute), 34
- KEY (*pandagg.node.query.full\_text.MultiMatch* attribute), 34
- KEY (*pandagg.node.query.full\_text.QueryString* attribute), 34
- KEY (*pandagg.node.query.full\_text.SimpleQueryString* attribute), 34
- KEY (*pandagg.node.query.geo.GeoBoundingBox* attribute), 34
- KEY (*pandagg.node.query.geo.GeoDistance* attribute), 34
- KEY (*pandagg.node.query.geo.GeoPolygone* attribute), 34
- KEY (*pandagg.node.query.geo.GeoShape* attribute), 34
- KEY (*pandagg.node.query.joining.HasChild* attribute), 34
- KEY (*pandagg.node.query.joining.HasParent* attribute), 35
- KEY (*pandagg.node.query.joining.Nested* attribute), 35
- KEY (*pandagg.node.query.joining.ParentId* attribute), 35
- KEY (*pandagg.node.query.shape.Shape* attribute), 35
- KEY (*pandagg.node.query.specialized.DistanceFeature* attribute), 35
- KEY (*pandagg.node.query.specialized.MoreLikeThis* attribute), 35
- KEY (*pandagg.node.query.specialized.Percolate* attribute), 35
- KEY (*pandagg.node.query.specialized.RankFeature* attribute), 36
- KEY (*pandagg.node.query.specialized.Script* attribute), 36
- KEY (*pandagg.node.query.specialized.Wrapper* attribute), 36
- KEY (*pandagg.node.query.specialized\_compound.PinnedQuery* attribute), 36
- KEY (*pandagg.node.query.specialized\_compound.ScriptScore* attribute), 36
- KEY (*pandagg.node.query.term\_level.Exists* attribute), 36
- KEY (*pandagg.node.query.term\_level.Fuzzy* attribute), 36
- KEY (*pandagg.node.query.term\_level.Ids* attribute), 36
- KEY (*pandagg.node.query.term\_level.Prefix* attribute), 37
- KEY (*pandagg.node.query.term\_level.Range* attribute), 37
- KEY (*pandagg.node.query.term\_level.Regexp* attribute), 37
- KEY (*pandagg.node.query.term\_level.Term* attribute), 37
- KEY (*pandagg.node.query.term\_level.Terms* attribute), 37
- KEY (*pandagg.node.query.term\_level.TermsSet* attribute), 37
- KEY (*pandagg.node.query.term\_level.Type* attribute), 37
- KEY (*pandagg.node.query.term\_level.Wildcard* attribute), 37
- KEY (*pandagg.query.Bool* attribute), 58
- KEY (*pandagg.query.Boosting* attribute), 58
- KEY (*pandagg.query.Common* attribute), 58
- KEY (*pandagg.query.ConstantScore* attribute), 58
- KEY (*pandagg.query.DisMax* attribute), 59
- KEY (*pandagg.query.DistanceFeature* attribute), 60
- KEY (*pandagg.query.Exists* attribute), 56
- KEY (*pandagg.query.FunctionScore* attribute), 59
- KEY (*pandagg.query.Fuzzy* attribute), 56
- KEY (*pandagg.query.GeoBoundingBox* attribute), 60
- KEY (*pandagg.query.GeoDistance* attribute), 60
- KEY (*pandagg.query.GeoPolygone* attribute), 60
- KEY (*pandagg.query.GeoShape* attribute), 59
- KEY (*pandagg.query.HasChild* attribute), 59
- KEY (*pandagg.query.HasParent* attribute), 59
- KEY (*pandagg.query.Ids* attribute), 56
- KEY (*pandagg.query.Intervals* attribute), 57
- KEY (*pandagg.query.Match* attribute), 57
- KEY (*pandagg.query.MatchBoolPrefix* attribute), 58
- KEY (*pandagg.query.MatchPhrase* attribute), 58
- KEY (*pandagg.query.MatchPhrasePrefix* attribute), 58
- KEY (*pandagg.query.MoreLikeThis* attribute), 60
- KEY (*pandagg.query.MultiMatch* attribute), 58
- KEY (*pandagg.query.Nested* attribute), 59
- KEY (*pandagg.query.ParentId* attribute), 59
- KEY (*pandagg.query.Percolate* attribute), 60
- KEY (*pandagg.query.PinnedQuery* attribute), 61
- KEY (*pandagg.query.Prefix* attribute), 57
- KEY (*pandagg.query.QueryString* attribute), 58
- KEY (*pandagg.query.Range* attribute), 57
- KEY (*pandagg.query.RankFeature* attribute), 60
- KEY (*pandagg.query.Regexp* attribute), 57
- KEY (*pandagg.query.Script* attribute), 60
- KEY (*pandagg.query.ScriptScore* attribute), 60
- KEY (*pandagg.query.Shape* attribute), 59
- KEY (*pandagg.query.SimpleQueryString* attribute), 58
- KEY (*pandagg.query.Term* attribute), 57
- KEY (*pandagg.query.Terms* attribute), 57

- KEY (*pandagg.query.TermsSet* attribute), 57
- KEY (*pandagg.query.Type* attribute), 57
- KEY (*pandagg.query.Wildcard* attribute), 57
- KEY (*pandagg.query Wrapper* attribute), 60
- KEY\_SEP (*pandagg.agg.Range* attribute), 45
- KEY\_SEP (*pandagg.node.agg.bucket.DateRange* attribute), 20
- KEY\_SEP (*pandagg.node.agg.bucket.Range* attribute), 21
- Keyword (*class in pandagg.mapping*), 51
- Keyword (*class in pandagg.node.mapping.field\_datatypes*), 28
- L**
- LeafQueryClause (*class in pandagg.node.query.abstract*), 31
- list\_documents() (*pandagg.interactive.response.IResponse* method), 16
- list\_nesteds\_at\_field() (*pandagg.mapping.Mapping* method), 50
- list\_nesteds\_at\_field() (*pandagg.tree.mapping.Mapping* method), 40
- Long (*class in pandagg.mapping*), 51
- Long (*class in pandagg.node.mapping.field\_datatypes*), 28
- LongRange (*class in pandagg.mapping*), 52
- LongRange (*class in pandagg.node.mapping.field\_datatypes*), 28
- M**
- MapperAnnotatedText (*class in pandagg.mapping*), 53
- MapperAnnotatedText (*class in pandagg.node.mapping.field\_datatypes*), 28
- MapperMurMur3 (*class in pandagg.mapping*), 53
- MapperMurMur3 (*class in pandagg.node.mapping.field\_datatypes*), 28
- Mapping (*class in pandagg.mapping*), 50
- Mapping (*class in pandagg.tree.mapping*), 40
- mapping\_type\_of\_field() (*pandagg.mapping.Mapping* method), 50
- mapping\_type\_of\_field() (*pandagg.tree.mapping.Mapping* method), 40
- MappingError, 50
- Match (*class in pandagg.node.query.full\_text*), 33
- Match (*class in pandagg.query*), 57
- MatchAll (*class in pandagg.agg*), 44
- MatchAll (*class in pandagg.node.agg.bucket*), 20
- MatchBoolPrefix (*class in pandagg.node.query.full\_text*), 33
- MatchBoolPrefix (*class in pandagg.query*), 57
- MatchPhrase (*class in pandagg.node.query.full\_text*), 33
- MatchPhrase (*class in pandagg.query*), 58
- MatchPhrasePrefix (*class in pandagg.node.query.full\_text*), 34
- MatchPhrasePrefix (*class in pandagg.query*), 58
- Max (*class in pandagg.agg*), 46
- Max (*class in pandagg.node.agg.metric*), 22
- MaxBucket (*class in pandagg.agg*), 48
- MaxBucket (*class in pandagg.node.agg.pipeline*), 24
- Meta (*class in pandagg.mapping*), 55
- Meta (*class in pandagg.node.mapping.meta\_fields*), 30
- MetricAgg (*class in pandagg.node.agg.abstract*), 18
- Min (*class in pandagg.agg*), 46
- Min (*class in pandagg.node.agg.metric*), 22
- MinBucket (*class in pandagg.agg*), 48
- MinBucket (*class in pandagg.node.agg.pipeline*), 24
- Missing (*class in pandagg.node.agg.bucket*), 20
- MoreLikeThis (*class in pandagg.node.query.specialized*), 35
- MoreLikeThis (*class in pandagg.query*), 60
- MovingAvg (*class in pandagg.agg*), 48
- MovingAvg (*class in pandagg.node.agg.pipeline*), 24
- MultiFieldsQueryClause (*class in pandagg.node.query.abstract*), 31
- MultiMatch (*class in pandagg.node.query.full\_text*), 34
- MultiMatch (*class in pandagg.query*), 58
- MultipleBucketAgg (*class in pandagg.node.agg.abstract*), 18
- must() (*pandagg.query.Query* method), 56
- must() (*pandagg.tree.query.Query* method), 41
- must\_not() (*pandagg.query.Query* method), 56
- must\_not() (*pandagg.tree.query.Query* method), 41
- N**
- name (*pandagg.node.query.abstract.QueryClause* attribute), 31
- Nested (*class in pandagg.agg*), 46
- Nested (*class in pandagg.mapping*), 52
- Nested (*class in pandagg.node.agg.bucket*), 21
- Nested (*class in pandagg.node.mapping.field\_datatypes*), 28
- Nested (*class in pandagg.node.query.joining*), 35
- Nested (*class in pandagg.query*), 59
- nested() (*pandagg.query.Query* method), 56
- nested() (*pandagg.tree.query.Query* method), 41
- nested\_at\_field() (*pandagg.mapping.Mapping* method), 50
- nested\_at\_field() (*pandagg.tree.mapping.Mapping* method), 40

40  
 NID\_SIZE (*pandagg.node.query.abstract.QueryClause attribute*), 31  
 node\_class (*pandagg.agg.Agg attribute*), 44  
 node\_class (*pandagg.mapping.Mapping attribute*), 50  
 node\_class (*pandagg.query.Query attribute*), 56  
 node\_class (*pandagg.tree.agg.Agg attribute*), 39  
 node\_class (*pandagg.tree.mapping.Mapping attribute*), 40  
 node\_class (*pandagg.tree.query.Query attribute*), 41  
 node\_path() (*pandagg.mapping.Mapping method*), 50  
 node\_path() (*pandagg.tree.mapping.Mapping method*), 40

## O

Obj (*class in pandagg.interactive.abstract*), 15  
 Object (*class in pandagg.mapping*), 52  
 Object (*class in pandagg.node.mapping.field\_datatypes*), 29  
 operator() (*pandagg.node.query.compound.CompoundClause class method*), 32  
 ordered() (*in module pandagg.utils*), 61

## P

pandagg (*module*), 61  
 pandagg.agg (*module*), 43  
 pandagg.client (*module*), 49  
 pandagg.exceptions (*module*), 49  
 pandagg.interactive (*module*), 17  
 pandagg.interactive.abstract (*module*), 15  
 pandagg.interactive.client (*module*), 16  
 pandagg.interactive.index (*module*), 16  
 pandagg.interactive.mapping (*module*), 16  
 pandagg.interactive.response (*module*), 16  
 pandagg.mapping (*module*), 50  
 pandagg.node (*module*), 38  
 pandagg.node.agg (*module*), 25  
 pandagg.node.agg.abstract (*module*), 17  
 pandagg.node.agg.bucket (*module*), 19  
 pandagg.node.agg.deserializer (*module*), 22  
 pandagg.node.agg.metric (*module*), 22  
 pandagg.node.agg.pipeline (*module*), 23  
 pandagg.node.mapping (*module*), 31  
 pandagg.node.mapping.abstract (*module*), 25  
 pandagg.node.mapping.deserializer (*module*), 26  
 pandagg.node.mapping.field\_datatypes (*module*), 26  
 pandagg.node.mapping.meta\_fields (*module*), 30  
 pandagg.node.mixins (*module*), 38  
 pandagg.node.query (*module*), 37  
 pandagg.node.query.abstract (*module*), 31

pandagg.node.query.compound (*module*), 32  
 pandagg.node.query.deserializer (*module*), 33  
 pandagg.node.query.full\_text (*module*), 33  
 pandagg.node.query.geo (*module*), 34  
 pandagg.node.query.joining (*module*), 34  
 pandagg.node.query.shape (*module*), 35  
 pandagg.node.query.span (*module*), 35  
 pandagg.node.query.specialized (*module*), 35  
 pandagg.node.query.specialized\_compound (*module*), 36  
 pandagg.node.query.term\_level (*module*), 36  
 pandagg.node.response (*module*), 38  
 pandagg.node.response.bucket (*module*), 37  
 pandagg.node.types (*module*), 38  
 pandagg.query (*module*), 55  
 pandagg.tree (*module*), 43  
 pandagg.tree.agg (*module*), 38  
 pandagg.tree.mapping (*module*), 40  
 pandagg.tree.query (*module*), 41  
 pandagg.tree.response (*module*), 42  
 pandagg.utils (*module*), 61  
 params() (*pandagg.node.query.compound.CompoundClause class method*), 32  
 PARAMS\_WHITELIST (*pandagg.node.query.compound.Bool attribute*), 32  
 PARAMS\_WHITELIST (*pandagg.node.query.compound.Boosting attribute*), 32  
 PARAMS\_WHITELIST (*pandagg.node.query.compound.CompoundClause attribute*), 32  
 PARAMS\_WHITELIST (*pandagg.node.query.compound.ConstantScore attribute*), 33  
 PARAMS\_WHITELIST (*pandagg.node.query.compound.DisMax attribute*), 33  
 PARAMS\_WHITELIST (*pandagg.node.query.compound.FunctionScore attribute*), 33  
 PARAMS\_WHITELIST (*pandagg.node.query.joining.HasChild attribute*), 35  
 PARAMS\_WHITELIST (*pandagg.node.query.joining.HasParent attribute*), 35  
 PARAMS\_WHITELIST (*pandagg.node.query.joining.Nested attribute*), 35  
 PARAMS\_WHITELIST (*pandagg.node.query.specialized\_compound.Pinned attribute*), 36  
 PARAMS\_WHITELIST (*pandagg.node.query.specialized\_compound.Script attribute*), 36  
 PARAMS\_WHITELIST (*pandagg.query.Bool attribute*), 58  
 PARAMS\_WHITELIST (*pandagg.query.Boosting attribute*), 58  
 PARAMS\_WHITELIST (*pandagg.query.ConstantScore attribute*), 59  
 PARAMS\_WHITELIST (*pandagg.query.DisMax attribute*), 59



tribute), 59

PARAMS\_WHITELIST (pandagg.query.FunctionScore attribute), 59

PARAMS\_WHITELIST (pandagg.query.HasChild attribute), 59

PARAMS\_WHITELIST (pandagg.query.HasParent attribute), 59

PARAMS\_WHITELIST (pandagg.query.Nested attribute), 59

PARAMS\_WHITELIST (pandagg.query.PinnedQuery attribute), 61

PARAMS\_WHITELIST (pandagg.query.ScriptScore attribute), 60

parent\_id() (pandagg.query.Query method), 56

parent\_id() (pandagg.tree.query.Query method), 41

ParentId (class in pandagg.node.query.joining), 35

ParentId (class in pandagg.query), 59

parse\_aggregation() (pandagg.tree.response.ResponseTree method), 42

paste() (pandagg.agg.Agg method), 44

paste() (pandagg.tree.agg.Agg method), 39

PercentileRanks (class in pandagg.agg), 47

PercentileRanks (class in pandagg.node.agg.metric), 23

Percentiles (class in pandagg.agg), 47

Percentiles (class in pandagg.node.agg.metric), 23

PercentilesBucket (class in pandagg.agg), 48

PercentilesBucket (class in pandagg.node.agg.pipeline), 25

Percolate (class in pandagg.node.query.specialized), 35

Percolate (class in pandagg.query), 60

Percolator (class in pandagg.mapping), 53

Percolator (class in pandagg.node.mapping.field\_datatypes), 29

pinned\_query() (pandagg.query.Query method), 56

pinned\_query() (pandagg.tree.query.Query method), 41

PinnedQuery (class in pandagg.node.query.specialized\_compound), 36

PinnedQuery (class in pandagg.query), 61

Pipeline (class in pandagg.node.agg.abstract), 18

Prefix (class in pandagg.node.query.term\_level), 36

Prefix (class in pandagg.query), 57

pretty (pandagg.utils.PrettyNode attribute), 61

PrettyNode (class in pandagg.utils), 61

## Q

Query (class in pandagg.query), 55

Query (class in pandagg.tree.query), 41

query() (pandagg.agg.Agg method), 44

query() (pandagg.interactive.index.Index method), 16

query() (pandagg.query.Query method), 56

query() (pandagg.tree.agg.Agg method), 39

query() (pandagg.tree.query.Query method), 41

query\_dict() (pandagg.agg.Agg method), 44

query\_dict() (pandagg.node.agg.abstract.AggNode method), 17

query\_dict() (pandagg.query.Query method), 56

query\_dict() (pandagg.tree.agg.Agg method), 39

query\_dict() (pandagg.tree.query.Query method), 41

QueryClause (class in pandagg.node.query.abstract), 31

QueryString (class in pandagg.node.query.full\_text), 34

QueryString (class in pandagg.query), 58

## R

Range (class in pandagg.agg), 45

Range (class in pandagg.node.agg.bucket), 21

Range (class in pandagg.node.query.term\_level), 37

Range (class in pandagg.query), 57

RankFeature (class in pandagg.mapping), 53

RankFeature (class in pandagg.node.mapping.field\_datatypes), 29

RankFeature (class in pandagg.node.query.specialized), 35

RankFeature (class in pandagg.query), 60

RankFeatures (class in pandagg.mapping), 54

RankFeatures (class in pandagg.node.mapping.field\_datatypes), 29

Regex (class in pandagg.node.query.term\_level), 37

Regex (class in pandagg.query), 57

reset\_data() (pandagg.node.mapping.abstract.Field method), 25

ResponseTree (class in pandagg.tree.response), 42

ReverseNested (class in pandagg.agg), 46

ReverseNested (class in pandagg.node.agg.bucket), 21

ROOT\_NAME (pandagg.node.response.bucket.Bucket attribute), 37

Routing (class in pandagg.mapping), 55

Routing (class in pandagg.node.mapping.meta\_fields), 30

## S

ScaledFloat (class in pandagg.mapping), 51

ScaledFloat (class in pandagg.node.mapping.field\_datatypes), 29

Script (class in pandagg.node.query.specialized), 36

Script (class in pandagg.query), 60

- `script_score()` (*pandagg.query.Query* method), 56
  - `script_score()` (*pandagg.tree.query.Query* method), 41
  - `ScriptPipeline` (class in *pandagg.node.agg.abstract*), 19
  - `ScriptScore` (class in *pandagg.node.query.specialized\_compound*), 36
  - `ScriptScore` (class in *pandagg.query*), 60
  - `SearchAsYouType` (class in *pandagg.mapping*), 54
  - `SearchAsYouType` (class in *pandagg.node.mapping.field\_datatypes*), 29
  - `SerialDiff` (class in *pandagg.agg*), 49
  - `SerialDiff` (class in *pandagg.node.agg.pipeline*), 25
  - `serialize()` (*pandagg.mapping.Mapping* method), 50
  - `serialize()` (*pandagg.node.query.abstract.QueryClause* method), 31
  - `serialize()` (*pandagg.node.query.term\_level.Ids* method), 36
  - `serialize()` (*pandagg.query.Ids* method), 57
  - `serialize()` (*pandagg.tree.mapping.Mapping* method), 40
  - `serialize_response()` (*pandagg.agg.Agg* method), 44
  - `serialize_response()` (*pandagg.tree.agg.Agg* method), 40
  - `set_mapping()` (*pandagg.agg.Agg* method), 44
  - `set_mapping()` (*pandagg.interactive.index.Index* method), 16
  - `set_mapping()` (*pandagg.query.Query* method), 56
  - `set_mapping()` (*pandagg.tree.agg.Agg* method), 40
  - `set_mapping()` (*pandagg.tree.query.Query* method), 41
  - `ShadowRoot` (class in *pandagg.node.agg.abstract*), 19
  - `Shape` (class in *pandagg.mapping*), 54
  - `Shape` (class in *pandagg.node.mapping.field\_datatypes*), 29
  - `Shape` (class in *pandagg.node.query.shape*), 35
  - `Shape` (class in *pandagg.query*), 59
  - `Short` (class in *pandagg.mapping*), 51
  - `Short` (class in *pandagg.node.mapping.field\_datatypes*), 29
  - `SHORT_TAG` (*pandagg.node.query.abstract.SingleFieldQueryClause* attribute), 32
  - `SHORT_TAG` (*pandagg.node.query.full\_text.Match* attribute), 33
  - `SHORT_TAG` (*pandagg.node.query.full\_text.MatchBoolPrefix* attribute), 33
  - `SHORT_TAG` (*pandagg.node.query.full\_text.MatchPhrase* attribute), 34
  - `SHORT_TAG` (*pandagg.node.query.full\_text.MatchPhrasePrefix* attribute), 34
  - `SHORT_TAG` (*pandagg.node.query.term\_level.Term* attribute), 37
  - `SHORT_TAG` (*pandagg.query.Match* attribute), 57
  - `SHORT_TAG` (*pandagg.query.MatchBoolPrefix* attribute), 58
  - `SHORT_TAG` (*pandagg.query.MatchPhrase* attribute), 58
  - `SHORT_TAG` (*pandagg.query.MatchPhrasePrefix* attribute), 58
  - `SHORT_TAG` (*pandagg.query.Term* attribute), 57
  - `should()` (*pandagg.query.Query* method), 56
  - `should()` (*pandagg.tree.query.Query* method), 41
  - `show()` (*pandagg.mapping.Mapping* method), 50
  - `show()` (*pandagg.tree.mapping.Mapping* method), 40
  - `show()` (*pandagg.tree.response.ResponseTree* method), 42
  - `SimpleQueryString` (class in *pandagg.node.query.full\_text*), 34
  - `SimpleQueryString` (class in *pandagg.query*), 58
  - `SingleFieldQueryClause` (class in *pandagg.node.query.abstract*), 32
  - `Size` (class in *pandagg.mapping*), 55
  - `Size` (class in *pandagg.node.mapping.meta\_fields*), 31
  - `Source` (class in *pandagg.mapping*), 55
  - `Source` (class in *pandagg.node.mapping.meta\_fields*), 31
  - `SparseVector` (class in *pandagg.mapping*), 54
  - `SparseVector` (class in *pandagg.node.mapping.field\_datatypes*), 30
  - `Stats` (class in *pandagg.agg*), 46
  - `Stats` (class in *pandagg.node.agg.metric*), 23
  - `StatsBucket` (class in *pandagg.agg*), 48
  - `StatsBucket` (class in *pandagg.node.agg.pipeline*), 25
  - `Sum` (class in *pandagg.agg*), 46
  - `Sum` (class in *pandagg.node.agg.metric*), 23
  - `SumBucket` (class in *pandagg.agg*), 48
  - `SumBucket` (class in *pandagg.node.agg.pipeline*), 25
- ## T
- `tag` (*pandagg.node.agg.abstract.AggNode* attribute), 18
  - `tag` (*pandagg.node.agg.abstract.ShadowRoot* attribute), 19
  - `tag` (*pandagg.node.query.abstract.MultiFieldsQueryClause* attribute), 31
  - `tag` (*pandagg.node.query.abstract.QueryClause* attribute), 31
  - `tag` (*pandagg.node.query.abstract.SingleFieldQueryClause* attribute), 32
  - `tag` (*pandagg.node.query.geo.GeoDistance* attribute), 34
  - `tag` (*pandagg.node.query.term\_level.Exists* attribute), 36
  - `tag` (*pandagg.node.query.term\_level.Ids* attribute), 36
  - `tag` (*pandagg.node.query.term\_level.Terms* attribute), 37
  - `tag` (*pandagg.query.Exists* attribute), 56

- tag (*pandagg.query.GeoDistance attribute*), 60
  - tag (*pandagg.query.Ids attribute*), 57
  - tag (*pandagg.query.Terms attribute*), 57
  - Term (*class in pandagg.node.query.term\_level*), 37
  - Term (*class in pandagg.query*), 57
  - Terms (*class in pandagg.agg*), 44
  - Terms (*class in pandagg.node.agg.bucket*), 21
  - Terms (*class in pandagg.node.query.term\_level*), 37
  - Terms (*class in pandagg.query*), 57
  - TermsSet (*class in pandagg.node.query.term\_level*), 37
  - TermsSet (*class in pandagg.query*), 57
  - Text (*class in pandagg.mapping*), 51
  - Text (*class in pandagg.node.mapping.field\_datatypes*), 30
  - to\_key (*pandagg.agg.Range attribute*), 45
  - to\_key (*pandagg.node.agg.bucket.Range attribute*), 21
  - TokenCount (*class in pandagg.mapping*), 53
  - TokenCount (*class in pandagg.node.mapping.field\_datatypes*), 30
  - TopHits (*class in pandagg.agg*), 47
  - TopHits (*class in pandagg.node.agg.metric*), 23
  - tree\_repr (*pandagg.node.mapping.abstract.Field attribute*), 25
  - TreeBasedObj (*class in pandagg.interactive.abstract*), 15
  - Type (*class in pandagg.mapping*), 54
  - Type (*class in pandagg.node.mapping.meta\_fields*), 31
  - Type (*class in pandagg.node.query.term\_level*), 37
  - Type (*class in pandagg.query*), 57
- ## U
- UniqueBucketAgg (*class in pandagg.node.agg.abstract*), 19
- ## V
- valid\_on\_field\_type() (*pandagg.node.agg.abstract.AggregNode class method*), 18
  - valid\_on\_field\_type() (*pandagg.node.mixins.FieldValidityMixin class method*), 38
  - validate\_agg\_node() (*pandagg.mapping.Mapping method*), 51
  - validate\_agg\_node() (*pandagg.tree.mapping.Mapping method*), 41
  - validate\_tree() (*pandagg.agg.Aggreg method*), 44
  - validate\_tree() (*pandagg.tree.agg.Aggreg method*), 40
  - VALUE\_ATTRS (*pandagg.agg.Avg attribute*), 46
  - VALUE\_ATTRS (*pandagg.agg.AvgBucket attribute*), 48
  - VALUE\_ATTRS (*pandagg.agg.BucketScript attribute*), 49
  - VALUE\_ATTRS (*pandagg.agg.BucketSelector attribute*), 49
  - VALUE\_ATTRS (*pandagg.agg.BucketSort attribute*), 49
  - VALUE\_ATTRS (*pandagg.agg.Cardinality attribute*), 46
  - VALUE\_ATTRS (*pandagg.agg.CumulativeSum attribute*), 49
  - VALUE\_ATTRS (*pandagg.agg.DateHistogram attribute*), 45
  - VALUE\_ATTRS (*pandagg.agg.Derivative attribute*), 48
  - VALUE\_ATTRS (*pandagg.agg.ExtendedStats attribute*), 47
  - VALUE\_ATTRS (*pandagg.agg.ExtendedStatsBucket attribute*), 48
  - VALUE\_ATTRS (*pandagg.agg.Filter attribute*), 45
  - VALUE\_ATTRS (*pandagg.agg.Filters attribute*), 44
  - VALUE\_ATTRS (*pandagg.agg.GeoBound attribute*), 47
  - VALUE\_ATTRS (*pandagg.agg.GeoCentroid attribute*), 47
  - VALUE\_ATTRS (*pandagg.agg.Global attribute*), 45
  - VALUE\_ATTRS (*pandagg.agg.Histogram attribute*), 45
  - VALUE\_ATTRS (*pandagg.agg.Max attribute*), 46
  - VALUE\_ATTRS (*pandagg.agg.MaxBucket attribute*), 48
  - VALUE\_ATTRS (*pandagg.agg.Min attribute*), 46
  - VALUE\_ATTRS (*pandagg.agg.MinBucket attribute*), 48
  - VALUE\_ATTRS (*pandagg.agg.MovingAvg attribute*), 49
  - VALUE\_ATTRS (*pandagg.agg.Nested attribute*), 46
  - VALUE\_ATTRS (*pandagg.agg.PercentileRanks attribute*), 47
  - VALUE\_ATTRS (*pandagg.agg.Percentiles attribute*), 47
  - VALUE\_ATTRS (*pandagg.agg.PercentilesBucket attribute*), 48
  - VALUE\_ATTRS (*pandagg.agg.Range attribute*), 45
  - VALUE\_ATTRS (*pandagg.agg.ReverseNested attribute*), 46
  - VALUE\_ATTRS (*pandagg.agg.SerialDiff attribute*), 49
  - VALUE\_ATTRS (*pandagg.agg.Stats attribute*), 47
  - VALUE\_ATTRS (*pandagg.agg.StatsBucket attribute*), 48
  - VALUE\_ATTRS (*pandagg.agg.Sum attribute*), 46
  - VALUE\_ATTRS (*pandagg.agg.SumBucket attribute*), 48
  - VALUE\_ATTRS (*pandagg.agg.Terms attribute*), 44
  - VALUE\_ATTRS (*pandagg.agg.TopHits attribute*), 47
  - VALUE\_ATTRS (*pandagg.agg.ValueCount attribute*), 47
  - VALUE\_ATTRS (*pandagg.node.agg.abstract.AggregNode attribute*), 17
  - VALUE\_ATTRS (*pandagg.node.agg.abstract.BucketAggregNode attribute*), 18
  - VALUE\_ATTRS (*pandagg.node.agg.abstract.FieldOrScriptMetricAgg attribute*), 18
  - VALUE\_ATTRS (*pandagg.node.agg.abstract.MetricAgg attribute*), 18
  - VALUE\_ATTRS (*pandagg.node.agg.abstract.MultipleBucketAgg attribute*), 18
  - VALUE\_ATTRS (*pandagg.node.agg.abstract.Pipeline attribute*), 19

[VALUE\\_ATTRS \(pandagg.node.agg.abstract.ScriptPipeline attribute\), 19](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.abstract.UniqueBucket attribute\), 19](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.bucket.DateHistogram attribute\), 19](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.bucket.DateRange attribute\), 20](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.bucket.Filter attribute\), 20](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.bucket.Filters attribute\), 20](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.bucket.Global attribute\), 20](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.bucket.Histogram attribute\), 20](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.bucket.Missing attribute\), 21](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.bucket.Nested attribute\), 21](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.bucket.Range attribute\), 21](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.bucket.ReverseNested attribute\), 21](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.bucket.Terms attribute\), 21](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.metric.Avg attribute\), 22](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.metric.Cardinality attribute\), 22](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.metric.ExtendedStats attribute\), 22](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.metric.GeoBound attribute\), 22](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.metric.GeoCentroid attribute\), 22](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.metric.Max attribute\), 22](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.metric.Min attribute\), 22](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.metric.PercentileRanks attribute\), 23](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.metric.Percentiles attribute\), 23](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.metric.Stats attribute\), 23](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.metric.Sum attribute\), 23](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.metric.TopHits attribute\), 23](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.metric.ValueCount attribute\), 23](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.pipeline.AvgBucket attribute\), 23](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.pipeline.BucketScript attribute\), 24](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.pipeline.BucketSelector attribute\), 24](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.pipeline.BucketSort attribute\), 24](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.pipeline.CumulativeSum attribute\), 24](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.pipeline.Derivative attribute\), 24](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.pipeline.ExtendedStatsBucket attribute\), 24](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.pipeline.MaxBucket attribute\), 24](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.pipeline.MinBucket attribute\), 24](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.pipeline.MovingAvg attribute\), 25](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.pipeline.PercentilesBucket attribute\), 25](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.pipeline.SerialDiff attribute\), 25](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.pipeline.StatsBucket attribute\), 25](#)  
[VALUE\\_ATTRS \(pandagg.node.agg.pipeline.SumBucket attribute\), 25](#)  
[ValueCount \(class in pandagg.agg\), 47](#)  
[ValueCount \(class in pandagg.node.agg.metric\), 23](#)  
[VersionIncompatibilityError, 50](#)

## W

[WHITELISTED\\_MAPPING\\_TYPES \(pandagg.agg.Avg attribute\), 46](#)  
[WHITELISTED\\_MAPPING\\_TYPES \(pandagg.agg.DateHistogram attribute\), 45](#)  
[WHITELISTED\\_MAPPING\\_TYPES \(pandagg.agg.ExtendedStats attribute\), 47](#)  
[WHITELISTED\\_MAPPING\\_TYPES \(pandagg.agg.GeoBound attribute\), 47](#)  
[WHITELISTED\\_MAPPING\\_TYPES \(pandagg.agg.GeoCentroid attribute\), 47](#)  
[WHITELISTED\\_MAPPING\\_TYPES \(pandagg.agg.Histogram attribute\), 45](#)  
[WHITELISTED\\_MAPPING\\_TYPES \(pandagg.agg.Max attribute\), 46](#)  
[WHITELISTED\\_MAPPING\\_TYPES \(pandagg.agg.Min attribute\), 46](#)  
[WHITELISTED\\_MAPPING\\_TYPES \(pandagg.agg.Nested attribute\), 46](#)  
[WHITELISTED\\_MAPPING\\_TYPES \(pandagg.agg.PercentileRanks attribute\), 47](#)

WHITELISTED_MAPPING_TYPES ( <i>pandagg.agg.Percentiles</i> attribute), 47	( <i>pandagg.node.agg.metric.Stats</i> attribute), 23
WHITELISTED_MAPPING_TYPES ( <i>pandagg.agg.Range</i> attribute), 45	WHITELISTED_MAPPING_TYPES ( <i>pandagg.node.agg.metric.Sum</i> attribute), 23
WHITELISTED_MAPPING_TYPES ( <i>pandagg.agg.ReverseNested</i> attribute), 46	WHITELISTED_MAPPING_TYPES ( <i>pandagg.node.mixins.FieldValidityMixin</i> attribute), 38
WHITELISTED_MAPPING_TYPES ( <i>pandagg.agg.Stats</i> attribute), 47	Wildcard ( <i>class in pandagg.node.query.term_level</i> ), 37
WHITELISTED_MAPPING_TYPES ( <i>pandagg.agg.Sum</i> attribute), 46	Wildcard ( <i>class in pandagg.query</i> ), 57
WHITELISTED_MAPPING_TYPES ( <i>pandagg.node.agg.abstract.AggNode</i> attribute), 17	Wrapper ( <i>class in pandagg.node.query.specialized</i> ), 36
WHITELISTED_MAPPING_TYPES ( <i>pandagg.node.agg.bucket.DateHistogram</i> attribute), 19	Wrapper ( <i>class in pandagg.query</i> ), 60
WHITELISTED_MAPPING_TYPES ( <i>pandagg.node.agg.bucket.DateRange</i> attribute), 20	
WHITELISTED_MAPPING_TYPES ( <i>pandagg.node.agg.bucket.Histogram</i> attribute), 20	
WHITELISTED_MAPPING_TYPES ( <i>pandagg.node.agg.bucket.Nested</i> attribute), 21	
WHITELISTED_MAPPING_TYPES ( <i>pandagg.node.agg.bucket.Range</i> attribute), 21	
WHITELISTED_MAPPING_TYPES ( <i>pandagg.node.agg.bucket.ReverseNested</i> attribute), 21	
WHITELISTED_MAPPING_TYPES ( <i>pandagg.node.agg.metric.Avg</i> attribute), 22	
WHITELISTED_MAPPING_TYPES ( <i>pandagg.node.agg.metric.ExtendedStats</i> attribute), 22	
WHITELISTED_MAPPING_TYPES ( <i>pandagg.node.agg.metric.GeoBound</i> attribute), 22	
WHITELISTED_MAPPING_TYPES ( <i>pandagg.node.agg.metric.GeoCentroid</i> attribute), 22	
WHITELISTED_MAPPING_TYPES ( <i>pandagg.node.agg.metric.Max</i> attribute), 22	
WHITELISTED_MAPPING_TYPES ( <i>pandagg.node.agg.metric.Min</i> attribute), 22	
WHITELISTED_MAPPING_TYPES ( <i>pandagg.node.agg.metric.PercentileRanks</i> attribute), 23	
WHITELISTED_MAPPING_TYPES ( <i>pandagg.node.agg.metric.Percentiles</i> attribute), 23	
WHITELISTED_MAPPING_TYPES	