GoodData Pandas

Release 1.3.0

GoodData Corporation

Mar 10, 2023

CONTENTS:

GoodData Pandas contains a thin layer that utilizes GoodData Python SDK and allows you to conveniently create pandas series and data frames from the computations done against semantic model in your GoodData.CN workspace.

CHAPTER

ONE

INSTALLATION

1.1 Requirements

- Python 3.7 or newer
- GoodData.CN or GoodData Cloud

1.2 Installation

Run the following command to install the `gooddata-pandas` package on your system:

```
pip install gooddata-pandas
```

CHAPTER

TWO

EXAMPLES

Here are a couple of introductory examples how to create indexed and not-indexed series and data frames:

2.1 Series

```
from gooddata_pandas import GoodPandas

# GoodData.CN host in the form of uri eg. "http://localhost:3000"
host = "http://localhost:3000"
# GoodData.CN user token
token = "some_user_token"
# initialize the adapter to work on top of GD.CN host and use the provided␣
↪authentication token
gp = GoodPandas(host, token)

workspace_id = "demo"
series = gp.series(workspace_id)

# create indexed series
indexed_series = series.indexed(index_by="label/label_id", data_by="fact/measure_id")

# create non-indexed series containing just the values of measure sliced by elements of␣
↪the label
non_indexed = series.not_indexed(data_by="fact/measure_id", granularity="label/label_id")
```

2.2 Data Frames

```
from gooddata_pandas import GoodPandas

# GoodData.CN host in the form of uri eg. "http://localhost:3000"
host = "http://localhost:3000"
# GoodData.CN user token
token = "some_user_token"
# initialize the adapter to work on top of GD.CN host and use the provided␣
↪authentication token
gp = GoodPandas(host, token)
```

(continues on next page)

(continued from previous page)

```
workspace_id = "demo"
frames = gp.data_frames(workspace_id)

# create indexed data frame
indexed_df = frames.indexed(
    index_by="label/label_id",
    columns=dict(
        first_label='label/first_label_id',
        second_label='label/second_label_id',
        first_metric='metric/first_metric_id',
        second_metric='fact/fact_id'
    )
)

# create data frame with hierarchical index
indexed_df = frames.indexed(
    index_by=dict(first_label='label/first_label_id', second_label='label/second_label_id
↪'),
    columns=dict(first_metric='metric/first_metric_id', second_metric='fact/fact_id')
)

# create non-indexed data frame
non_indexed_df = frames.not_indexed(
    columns=dict(
        first_label='label/first_label_id',
        second_label='label/second_label_id',
        first_metric='metric/first_metric_id',
        second_metric='fact/fact_id'
    )
)

# create data frame based on the contents of the insight. if the insight contains labels␣
↪and
#  measures, the data frame will contain index or hierarchical index.
insight_df = frames.for_insight('insight_id')

# create data frame based on the content of the items dict. if the dict contains both␣
↪labels
# and measures, the frame will contain index or hierarchical index.
df = frames.for_items(
    items=dict(
        first_label='label/first_label_id',
        second_label='label/second_label_id',
        first_metric='metric/first_metric_id',
        second_metric='fact/fact_id'
    )
)

# create data frame from custom execution definition
exec_def = ExecutionDefinition(
    attributes=[
        Attribute(local_id="region", label="region"),
```

(continues on next page)

(continued from previous page)

```
        Attribute(local_id="state", label="state"),
        Attribute(local_id="product_category", label="products.category"),
    ],
    metrics=[
        SimpleMetric(local_id="price", item=ObjId(id="price", type="fact")),
        SimpleMetric(local_id="order_amount", item=ObjId(id="order_amount", type="metric
↪")),
    ],
    filters=[],
    dimensions=[["state", "region"], ["product_category", "measureGroup"]],
)
df, df_metadata = frames.for_exec_def(exec_def=exec_def)

# use result ID from computation above and generate dataframe just from it
df_from_result_id, df_metadata_from_result_id = frames.for_exec_result_id(
    result_id=df_metadata.execution_response.result_id,
)
```

CHAPTER

THREE

API

3.1 gooddata_pandas

Modules

3.1.1 gooddata_pandas.data_access

Functions

compute_and_extract(sdk, workspace_id, columns)	Convenience function to drive computation & data extraction on behalf of the series and data frame factories.

gooddata_pandas.data_access.compute_and_extract

gooddata_pandas.data_access.**compute_and_extract**(*sdk: GoodDataSdk*, *workspace_id: str*, *columns: ColumnsDef*, *index_by: Optional[IndexDef] = None*, *filter_by: Optional[Union[Filter, list[Filter]]] = None*) → tuple[dict, dict]

Convenience function to drive computation & data extraction on behalf of the series and data frame factories.

Given data columns and index columns, this function will create AFM execution and then read the results and populate data and index dicts.

For each column in *columns*, the returned data will contain key under which there is array of data for that column For each index in index_by, the returned data will contain key under which there is array with data to construct the index. When there are multiple indexes, feed the indexes to MultiIndex.from_arrays().

Note that as convenience it is possible to pass just single index. in that case the index dict will contain exactly one key of '0' (just get first value from dict when consuming the result).

Classes

ExecutionDefinitionBuilder(columns[, index_by])	

gooddata_pandas.data_access.ExecutionDefinitionBuilder

class gooddata_pandas.data_access.**ExecutionDefinitionBuilder**(*columns: Dict[str, Union[Attribute, Metric, ObjId, str]]*, *index_by: Optional[Union[Attribute, ObjId, str, Dict[str, Union[Attribute, ObjId, str]]]] = None*)

Bases: `object`

__init__(*columns: Dict[str, Union[Attribute, Metric, ObjId, str]]*, *index_by: Optional[Union[Attribute, ObjId, str, Dict[str, Union[Attribute, ObjId, str]]]] = None*) → None

Methods

__init__(columns[, index_by])	
build_execution_definition([filter_by])	

Attributes

col_to_attr_idx
col_to_metric_idx
index_to_attr_idx

3.1.2 gooddata_pandas.dataframe

Classes

DataFrameFactory(sdk, workspace_id)	Factory to create pandas.DataFrame instances.

gooddata_pandas.dataframe.DataFrameFactory

class gooddata_pandas.dataframe.**DataFrameFactory**(*sdk: GoodDataSdk*, *workspace_id: str*)

Bases: object

Factory to create pandas.DataFrame instances.

There are several methods in place that should provide for convenient construction of data frames:

- indexed() - calculate measure values sliced by one or more labels, indexed by those labels
- not_indexed() - calculate measure values sliced by one or more labels, but not indexed by those labels, label values will be part of the DataFrame and will be in the same row as the measure values calculated for them
- for_items() - calculate measure values for a one or more items which may be labels or measures. Depending what items you specify, this method will create DataFrame with or without index
- for_insight() - calculate DataFrame for insight created by GoodData.CN Analytical Designer. Depending on what items are in the insight, this method will create DataFrame with or without index.

Note that all of these methods have additional levels of convenience and flexibility so their purpose is not limited to just what is listed above.

__init__(*sdk: GoodDataSdk*, *workspace_id: str*) → None

Methods

__init__(sdk, workspace_id)	

for_exec_def(exec_def[, label_overrides, ...])	Creates a data frame using an execution definition.
for_exec_result_id(result_id[, ...])	Creates a data frame using an execution result's metadata identified by result_id.
for_insight(insight_id[, auto_index])	Creates a data frame with columns based on the content of the insight with the provided identifier.
for_items(items[, filter_by, auto_index])	Creates a data frame for a named items.
indexed(index_by, columns[, filter_by])	Creates a data frame indexed by values of the label.
not_indexed(columns[, filter_by])	Creates a data frame with columns created from metrics and or labels.
result_cache_metadata_for_exec_result_id(...)	Retrieves result cache metadata for given :result_id: :param result_id: ID of execution result to retrieve the metadata for :return: corresponding result cache metadata

for_exec_def(*exec_def: ExecutionDefinition*, *label_overrides: Optional[Dict[str, Dict[str, Dict[str, str]]]] = None*, *result_size_dimensions_limits: Tuple[Optional[int], ...] = ()*, *result_size_bytes_limit: Optional[int] = None*) → Tuple[DataFrame, *DataFrameMetadata*]

Creates a data frame using an execution definition. The data frame will respect the dimensionality specified in execution definition's result spec.

Each dimension may be sliced by multiple labels. The factory will create MultiIndex for the dataframe's row index and the columns.

Example of label_overrides structure:

```
{
    "labels": {
        "local_attribute_id": {
            "title": "My new attribute label"
        ,...
    },
    "metrics": {
        "local_metric_id": {
            "title": "My new metric label"
        },...
    }
}
```

Parameters

- **exec_def** – execution definition
- **label_overrides** – label overrides for metrics and attributes
- **result_size_dimensions_limits** – A tuple containing maximum size of result dimensions. Optional.
- **result_size_bytes_limits** – Maximum size of result in bytes. Optional.

Returns

tuple holding DataFrame and DataFrame metadata

for_exec_result_id(*result_id: str*, *label_overrides: Optional[Dict[str, Dict[str, Dict[str, str]]]] = None*, *result_cache_metadata: Optional[ResultCacheMetadata] = None*, *result_size_dimensions_limits: Tuple[Optional[int], ...] = ()*, *result_size_bytes_limit: Optional[int] = None*, *use_local_ids_in_headers: bool = False*) → Tuple[DataFrame, *DataFrameMetadata*]

Creates a data frame using an execution result's metadata identified by result_id. The data frame will respect the dimensionality specified in execution definition's result spec.

Each dimension may be sliced by multiple labels. The factory will create MultiIndex for the dataframe's row index and the columns.

Example of label_overrides structure:

```
{
    "labels": {
        "local_attribute_id": {
            "title": "My new attribute label"
        ,...
    },
    "metrics": {
        "local_metric_id": {
            "title": "My new metric label"
        },...
    }
}
```

Parameters

- **result_id** – executionResult ID from ExecutionResponse
- **label_overrides** – label overrides for metrics and attributes
- **result_cache_metadata** – Metadata for the corresponding exec result. Optional.
- **result_size_dimensions_limits** – A tuple containing maximum size of result dimensions. Optional.
- **result_size_bytes_limit** – Maximum size of result in bytes. Optional.
- **use_local_ids_in_headers** – Use local identifiers of header attributes and metrics. Optional.

Returns

tuple holding DataFrame and DataFrame metadata

for_insight(*insight_id: str*, *auto_index: bool = True*) → DataFrame

Creates a data frame with columns based on the content of the insight with the provided identifier. The filters that are set on the insight will be applied and used for the server-side computation of the data for the data frame.

This method will create DataFrame with or without index - depending on the contents of the insight. The rules are as follows:

- if the insight contains both attributes and measures, it will be mapped to a DataFrame with index
  - if there are multiple attributes, hieararchical index (pandas.MultiIndex) will be used
  - otherwise a normal index will be used (pandas.Index)
  - you can use the option 'auto_index' argument to disable this logic and force no indexing

- if the insight contains either only attributes or only measures, then DataFrame will not be indexed and all attribute or measures values will be used as data.

Note that if the insight consists of single measure only, the resulting data frame is guaranteed to have single 'row' of data with one column per measure.

Parameters

- **insight_id** – insight identifier
- **auto_index** – optionally force creation of DataFrame without index even if the data in the insight is eligible for indexing

Returns
pandas dataframe instance

for_items(*items: ColumnsDef*, *filter_by: Optional[Union[Filter, list[Filter]]] = None*, *auto_index: bool = True*) → pandas.DataFrame

Creates a data frame for a named items. This is a convenience method that will create DataFrame with or without index based on the context of the items that you pass.

- If items contain labels and measures, then DataFrame with index will be created. If there is more than one label among the items, then hierarchical index will be created.

  You can turn this behavior using 'auto_index' parameter.

- Otherwise DataFrame without index will be created and will contain column per item.

You may also optionally specify filters to apply during the computation on the server.

Parameters

- **items** – dict mapping item name to its definition; item may be specified as:
  - object identifier: `ObjId(id='some_id', type='<type>')` - where type is either `label`, `fact` or `metric`
  - string representation of object identifier: `<type>/some_id` - where type is either `label`, `fact` or `metric`
  - Attribute object used in the compute model: `Attribute(local_id=..., label='some_label_id')`
  - subclass of Measure object used in the compute model: SimpleMeasure, PopDateMeasure, PopDatasetMeasure, ArithmeticMeasure
- **filter_by** – optionally specify filters to apply during computation on the server, reference to filtering column can be one of:
  - string reference to item key
  - object identifier in string form
  - object identifier: `ObjId(id='some_label_id', type='<type>')`
  - Attribute or Metric depending on type of filter

Returns
pandas dataframe instance

indexed(*index_by: IndexDef*, *columns: ColumnsDef*, *filter_by: Optional[Union[Filter, list[Filter]]] = None*) → pandas.DataFrame

Creates a data frame indexed by values of the label. The data frame columns will be created from either metrics or other label values.

The computation to obtain data from GoodData.CN workspace will use all labels that you specify for both indexing and in columns to aggregate values of metric columns.

Note that depending on composition of the labels, the DataFrame's index may or may not be unique.

Parameters

- **index_by** – one or more labels to index by; specify either:
  - string with reference to columns key - only attribute can be referenced
  - string with id: `some_label_id`,
  - string representation of object identifier: `label/some_label_id`
  - object identifier: `ObjId(id='some_label_id', type='label')`,
  - or an Attribute object used in the compute model: `Attribute(local_id=..., label='some_label_id')`,
  - dict containing mapping of index name to label to use for indexing - specified in one of the ways list above
- **columns** – dict mapping column name to its definition; column may be specified as:
  - object identifier: `ObjId(id='some_id', type='<type>')` - where type is either `label`, `fact` or `metric`
  - string representation of object identifier: `<type>/some_id` - where type is either `label`, `fact` or `metric`
  - Attribute object used in the compute model: `Attribute(local_id=..., label='some_label_id')`
  - subclass of Measure object used in the compute model: SimpleMeasure, PopDateMeasure, PopDatasetMeasure, ArithmeticMeasure
- **filter_by** – optional filters to apply during computation on the server, reference to filtering column can be one of:
  - string reference to column key or index key
  - object identifier in string form
  - object identifier: `ObjId(id='some_label_id', type='<type>')`
  - Attribute or Metric depending on type of filter

Returns

pandas dataframe instance

not_indexed(*columns: ColumnsDef*, *filter_by: Optional[Union[Filter, list[Filter]]] = None*) → pandas.DataFrame

Creates a data frame with columns created from metrics and or labels.

The computation to obtain data from GoodData.CN workspace will use all labels that you specify for both columns to aggregate values of metric columns.

Parameters

- **columns** – dict mapping column name to its definition; column may be specified as:
  - object identifier: `ObjId(id='some_id', type='<type>')` - where type is either `label`, `fact` or `metric`
  - string representation of object identifier: `<type>/some_id` - where type is either `label`, `fact` or `metric`

  - Attribute object used in the compute model: `Attribute(local_id=..., label='some_label_id')`
  - subclass of Measure object used in the compute model: SimpleMeasure, PopDateMeasure, PopDatasetMeasure, ArithmeticMeasure
- **filter_by** – optionally specify filters to apply during computation on the server, reference to filtering column can be one of:
  - string reference to column key
  - object identifier in string form
  - object identifier: `ObjId(id='some_label_id', type='<type>')`
  - Attribute or Metric depending on type of filter

Returns
pandas dataframe instance

result_cache_metadata_for_exec_result_id(*result_id: str*) → ResultCacheMetadata

Retrieves result cache metadata for given :result_id: :param result_id: ID of execution result to retrieve the metadata for :return: corresponding result cache metadata

3.1.3 gooddata_pandas.good_pandas

Module Attributes

USER_AGENT	Extra segment of the User-Agent header that will be appended to standard gooddata-sdk user agent.

gooddata_pandas.good_pandas.USER_AGENT

gooddata_pandas.good_pandas.**USER_AGENT = 'gooddata-pandas/1.3.0'**

Extra segment of the User-Agent header that will be appended to standard gooddata-sdk user agent.

Classes

GoodPandas(host, token[, headers_host])	Facade to access factories that create pandas Series and DataFrames using analytics computed by GoodData.CN.

gooddata_pandas.good_pandas.GoodPandas

class gooddata_pandas.good_pandas.**GoodPandas**(*host: str*, *token: str*, *headers_host: Optional[str] = None*, ***custom_headers_: Optional[str]*)

Bases: `object`

Facade to access factories that create pandas Series and DataFrames using analytics computed by GoodData.CN.

__init__(*host: str*, *token: str*, *headers_host: Optional[str] = None*, ***custom_headers_: Optional[str]*) → None

Methods

__init__(host, token[, headers_host])	
create_from_profile([profile, profiles_path])	
data_frames(workspace_id)	Creates factory to use for construction of pandas.DataFrame.
series(workspace_id)	Creates factory to use for construction of pandas.Series.

Attributes

sdk

data_frames(*workspace_id: str*) → *DataFrameFactory*

Creates factory to use for construction of pandas.DataFrame.

Parameters
workspace_id – workspace to which the factory will be bound

Returns
always one same instance for given workspace

series(*workspace_id: str*) → *SeriesFactory*

Creates factory to use for construction of pandas.Series.

Parameters
workspace_id – workspace to which the factory will be bound

Returns
always one same instance for given workspace

3.1.4 gooddata_pandas.result_convertor

Functions

convert_execution_response_to_dataframe(...)	Converts execution result to a pandas dataframe, maintaining the dimensionality of the result.

gooddata_pandas.result_convertor.convert_execution_response_to_dataframe

gooddata_pandas.result_convertor.**convert_execution_response_to_dataframe**(*execution_response: BareExecutionResponse*, *result_cache_metadata: ResultCacheMetadata*, *label_overrides: Dict[str, Dict[str, Dict[str, str]]]*, *result_size_dimensions_limits: Tuple[Optional[int], ...]*, *result_size_bytes_limit: Optional[int] = None*, *use_local_ids_in_headers: bool = False*) → Tuple[DataFrame, *DataFrameMetadata*]

Converts execution result to a pandas dataframe, maintaining the dimensionality of the result.

Because the result itself does not contain all the necessary metadata to do the full conversion, this method expects that the execution _response_.

Parameters

- **label_overrides** – label overrides
- **execution_response** – execution response through which the result can be read and converted to a dataframe

Returns

a new dataframe

Classes

DataFrameMetadata(row_totals_indexes, ...)	

gooddata_pandas.result_convertor.DataFrameMetadata

class gooddata_pandas.result_convertor.**DataFrameMetadata**(*row_totals_indexes: List[List[int]]*, *execution_response: BareExecutionResponse*)

Bases: `object`

__init__(*row_totals_indexes: List[List[int]]*, *execution_response: BareExecutionResponse*) → None

Method generated by attrs for class DataFrameMetadata.

Methods

__init__(row_totals_indexes, execution_response)	Method generated by attrs for class DataFrameMetadata.
from_data(headers, execution_response)	

Attributes

row_totals_indexes
execution_response

3.1.5 gooddata_pandas.series

Classes

SeriesFactory(sdk, workspace_id)

gooddata_pandas.series.SeriesFactory

class gooddata_pandas.series.**SeriesFactory**(*sdk: GoodDataSdk*, *workspace_id: str*)

Bases: object

__init__(*sdk: GoodDataSdk*, *workspace_id: str*) → None

Methods

__init__(sdk, workspace_id)	
indexed(index_by, data_by[, filter_by])	Creates pandas Series from data points calculated from a single *data_by* that will be computed on granularity of the index labels.
not_indexed(data_by[, granularity, filter_by])	Creates pandas Series from data points calculated from a single *data_by* that will be computed on granularity of the specified labels.

indexed(*index_by: IndexDef*, *data_by: Union[SimpleMetric, str, ObjId, Attribute]*, *filter_by: Optional[Union[Filter, list[Filter]]] = None*) → pandas.Series

Creates pandas Series from data points calculated from a single *data_by* that will be computed on granularity of the index labels. The elements of the index labels will be used to construct simple or hierarchical index.

Parameters

- **index_by** – label to index by; specify either:
  - string with id: `some_label_id`,
  - object identifier: `ObjId(id='some_label_id', type='label')`,
  - string representation of object identifier: `label/some_label_id`
  - or an Attribute object used in the compute model: `Attribute(local_id=..., label='some_label_id')`
  - dict containing mapping of index name to label to use for indexing - specified in one of the ways list above
- **data_by** – label, fact or metric to that will provide data (metric values or label elements); specify either:
  - object identifier: `ObjId(id='some_id', type='<type>')` - where type is either `label`, `fact` or `metric`
  - string representation of object identifier: `<type>/some_id` - where type is either `label`, `fact` or `metric`
  - Attribute object used in the compute model: `Attribute(local_id=..., label='some_label_id')`
  - SimpleMetric object used in the compute model: `SimpleMetric(local_id=..., item=..., aggregation=...)`
- **filter_by** – optionally specify filter to apply during computation on the server, reference to filtering column can be one of:
  - string reference to index key
  - object identifier in string form
  - object identifier: `ObjId(id='some_label_id', type='<type>')`
  - Attribute or Metric depending on type of filter

Returns

pandas series instance

not_indexed(*data_by: Union[SimpleMetric, str, ObjId, Attribute]*, *granularity: Optional[Union[list[LabelItemDef], IndexDef]] = None*, *filter_by: Optional[Union[Filter, list[Filter]]] = None*) → pandas.Series

Creates pandas Series from data points calculated from a single *data_by* that will be computed on granularity of the specified labels. No index will be constructed.

Note that data_by may also be a label in which case the Series will contain label elements.

Parameters

- **data_by** – label, fact or metric to get data from; specify either:
  - object identifier: `ObjId(id='some_id', type='<type>')` - where type is either `label`, `fact` or `metric`
  - string representation of object identifier: `<type>/some_id` - where type is either `label`, `fact` or `metric`
  - Attribute object used in the compute model: `Attribute(local_id=..., label='some_label_id')`
  - SimpleMetric object used in the compute model: `SimpleMetric(local_id=..., item=..., aggregation=...)`

- **granularity** – optionally specify label to slice the metric by; specify either:
  - string with id: `some_label_id`,
  - object identifier: `ObjId(id='some_label_id', type='label')`,
  - string representation of object identifier: `label/some_label_id`
  - or an Attribute object used in the compute model: `Attribute(local_id=..., label='some_label_id')`
  - list containing multiple labels to slice the metric by - specified in one of the ways list above
  - dict containing mapping of index name to label to use for indexing - specified in one of the ways list above; this option is available so that you can easily switch from indexed factory method to this one if needed
- **filter_by** – optionally specify filter to apply during computation on the server, reference to filtering column can be one of:
  - object identifier in string form
  - object identifier: `ObjId(id='some_label_id', type='<type>')`
  - Attribute or Metric depending on type of filter

Returns
pandas series instance

3.1.6 gooddata_pandas.utils

Functions

make_pandas_index(index)

gooddata_pandas.utils.make_pandas_index

gooddata_pandas.utils.**make_pandas_index**(*index: dict*) → Optional[Union[Index, MultiIndex]]

Classes

DefaultInsightColumnNaming()

gooddata_pandas.utils.DefaultInsightColumnNaming

class gooddata_pandas.utils.**DefaultInsightColumnNaming**

Bases: object

__init__() → None

Methods

__init__()
col_name_for_attribute(attr)
col_name_for_metric(measure)

PYTHON MODULE INDEX

g

INDEX

Symbols

C

D

E

F

G

I

M

N

R

S

U