Datasets

Dataset files define datasets to use in the repository. Each dataset file in your repository must correspond to either a physical table/view in your database, or the results of a `SELECT` statement.

Note: Dataset files must define **all** columns in the physical tables they reference, and can therefore be quite large. Because of this, AtScale recommends sharing these files across repositories.

Dataset files support the following properties.

Unique_name

- **Type:** string
- **Required:** Y

The unique name of the dataset. This must be unique across all repositories and subrepositories.

Object_type

- **Type:** const
- **Required:** Y

The type of object defined by the file. For datasets, the value of this property must be `dataset`.

Label

- **Type:** string
- **Required:** Y

The name of the dataset, as it appears in AtScale. This value does not need to be unique.

Connection_id

- **Type:** string
- **Required:** Y

The `unique_name` of the connection object that defines the database and schema in which the dataset is stored.

Sql

- **Type:** string

- **Required:** Required if `table` is not defined

A SQL query used to pull data from a specific connection defined within the repository, similar to a database view. This determines whether the dataset file defines a query dataset.

Table

- **Type:** string
- **Required:** Required if `sql` is not defined

The name of the table in the database that the dataset is based on.

Columns

- **Type:** array
- **Required:** Y

Defines the columns available in the dataset.

Note: You should define **all** columns available in the dataset. This is especially important for dataset files that are shared across multiple repositories.

The `columns` property within a dataset file supports the following properties.

Name

- **Type:** string
- **Required:** Y

The name of the column.

Data_type

- **Type:** string
- **Required:** Required unless this column is a `map`

The data type of the values within the column.

Supported values:

- `string`

- int
- long
- bigint
- tinyint
- float
- double
- decimal
- decimal(x,y)
- number
- number(x,y)
- numeric(x,y)
- boolean
- date
- datetime
- timestamp

Sql

- **Type:** string
- **Required:** N

Defines the column as a calculated column.

Calculated columns enable you to add simple data transformations to the dataset. These can be used as the basis of model attributes, just like any other dataset column. For more information, see Adding Calculated Columns To Datasets For Simple Data Transformations.

The value of this property should be a valid SQL statement that can be run as part of the `SELECT` list of a query.

The SQL statement is passed directly to the underlying database when the query runs, so it must be in a syntax that is supported by your chosen engine. If you want to run the query on other types of databases, use the `dialects` property to define additional dialects for it to run in.

Dialects

- **Type:** array
- **Required:** N

Defines alternate dialects for the `sql` statement so that it can run on other types of databases. You can define as many alternate dialects as needed.

Supported properties:

- `dialect`: String, required. An alternate SQL dialect.
  Supported values:
  - `Snowflake`
  - `Postgresql`
  - `DatabricksSQL`
  - `BigQuery`
  - `Iris`
- `sql`: String, required. The alternate SQL statement.

Map

- **Type:** object
- **Required:** N

Defines a map to use to create a calculated column.

Supported properties:

- `field_terminator`: String, required. The delimiter used to separate the key:value pairs. This must be in quotes (").
- `key_terminator`: String, required. The delimiter used to separate the individual keys from their values. This must be in quotes (").
- `key_type`: String, required. The data type of the map's keys.
- `value_type`: String, required. The data type of the map's values.

The mapped columns are defined as separate columns within the dataset file. Each of these must have the `parent_column` property.

For more information on maps in AtScale, see Extract Values From A Map.

Parent_column

- **Type:** string
- **Required:** Required for mapped columns

For mapped columns only. Specifies the `map` column used to create this column.

Description

- **Type:** string
- **Required:** N

A description of the dataset.

Incremental

- **Type:** object
- **Required:** N

Enables incremental builds for the dataset. When the engine performs an incremental rebuild of an aggregate table, it appends new rows and updates existing rows that fall within a specified period of time (called the grace period). For more information, see About Incremental Rebuilds.

The `incremental` property supports the following properties.

Column

- **Type:** string
- **Required:** Y

The name of the dataset column to use as an incremental indicator. This column must have values that increase monotonically, such as a numeric UNIX timestamp showing seconds since epoch, or a Timestamp/DateTime. The values in this column enable the AtScale engine both to append rows to an aggregate table and update rows during an incremental rebuild.

The values of this column should be of one of the following data types: Long, Integer, Decimal (38,0) (Snowflake only), Timestamp, DateTime.

If you do not have a column that meets this criteria, you may need to create a calculated column to transform string or datetime values into the right data type.

Grace_period

- **Type:** string
- **Required:** Y

When the AtScale engine starts an incremental build, the `grace_period` determines how far back in time the engine looks for updates to rows in the dataset; for example, one week or 15 days.

The value of this property should be an integer, followed by the time unit. The time unit can be any of the following: `s` (second), `m` (minute), `h` (hour), `d` (day), `w` (week).

For example, setting the value to `'100s'` sets the grace period to 100 seconds. Setting it to `'1w'` sets the grace period to one week.

Immutable

- **Type:** boolean
- **Required:** N

Determines whether the dataset changes often or not. The AtScale engine uses this information when running incremental builds of aggregates that use joins on dimensions that do not change often.