YETL

Yet Another ETL Framework

Python API

The goal of yetl is to easily declare data pipeline configuration. Once declared you'll want to access that data strcutures that you've declared using python. This reference shows the canonical patterns to load up the configuration and use it however you see fit.

Initilise

Each time you want to access the configuration using the API you will need to deserialize the configuration into python objects held in memory.

Load Configuration

The following example loads the configuration from the following configuration files into an instance of the Config object:

- ./my_project/pipelines/tables.yaml
- ./my_project/pipelines/batch.yaml

```
from yetl import (
  Config, StageType
)

config = Config(
  project="my_project",
  pipeline="batch"
)
```

The full Config constructor has the following arguments:

- `project: str`

Name for the project.

- `pipeline: str`

Name of the pipeline config file without the extension that has the config you want to use.

- `timeslice: Timeslice = None`

The timeslice that you want to inject into the config and replace the date time mask jinja variables in the configuration (see the Timeslice section). The timeslice is optional since you may just want to pull back a collection of delta tables for operations other than loading data or load the current UTC datetime which is the default.

- `config_path: str = None`

Yetl will do it's best to figure out where the configuration is located based on your project file confg, operating env and the project configuration file. If you're not using the standard settings for whatever reason you can provide an explicit path to where youtr configuration resides.

Inject Timeslice

Yetl provides a timeslice object for the convenience of injecting time periods into custom formats on data file paths and file names. For example it's typical to land data files partitioned as follows:

```
/Volumes/development/my_project/customer/2023/07/30/customer-20230730.json
```

In the pipeline configuration we generalise this by declaring the following expression definition. This tells yetl where to insert the timeslice using what datetime format. The datatime format is expressed as a python datetime format. There are 2 formats because the filename and path datetime format can be different:

```
    location: "/Volumes/{{catalog}}/my_project/{{table}}/{{path_date_format}}"
    filename: "{{table}}-{{filename_date_format}}*.json"
    filename_date_format: "%Y%m%d"
    path_date_format: "%Y/%m/%d"
```

In some loading patterns we need to inject the period of data we want to load into the config. The `Timeslice` object is provided specifically to do this since it internally handles datetime formatting, validation and wildcard handling for bulk data loading.

Specific Day

For example loading a **specific day**, injecting this `Timeslice`:

```
from yetl import (
  Config, Timeslice
)

config = Config(
  project = "my_project",
  timeslice = Timeslice(year=2023, month=7, day=30)
  pipeline = "batch"
)
```

Will result in this path:

```
/Volumes/development/my_project/customer/2023/07/30/customer-20230730*.json
```

Partial Bulk

For example bulk **loading a year**, injecting this `Timeslice`:

```
from yetl import (
  Config, Timeslice
)

config = Config(
  project = "my_project",
  timeslice = Timeslice(year=2023, month="*", day="*")
  pipeline = "batch"
)
```

Will result in this wildcard path:

```
/Volumes/development/my_project/customer/2023/*/*/customer-2023***.json
```

Full Bulk

For example **all time**, injecting this `Timeslice`:

```
from yetl import (
  Config, Timeslice
)

config = Config(
  project = "my_project",
  timeslice = Timeslice(year="*", month="*", day="*")
  pipeline = "batch"
)
```

Will result in this wildcard path:

```
/Volumes/development/my_project/customer/*/*/*/customer-****.json
```

Note:

If you're stream loading using databricks cloud files and trigger now, you don't need to worry about timeslice loading your data since databricks will automatically track and checkpoint the files that you're loading. However batch stream loading also has some downsides, but not worry since yetl has you covered making it easy to inject timeslice loading.

Yet Another ETL Framework