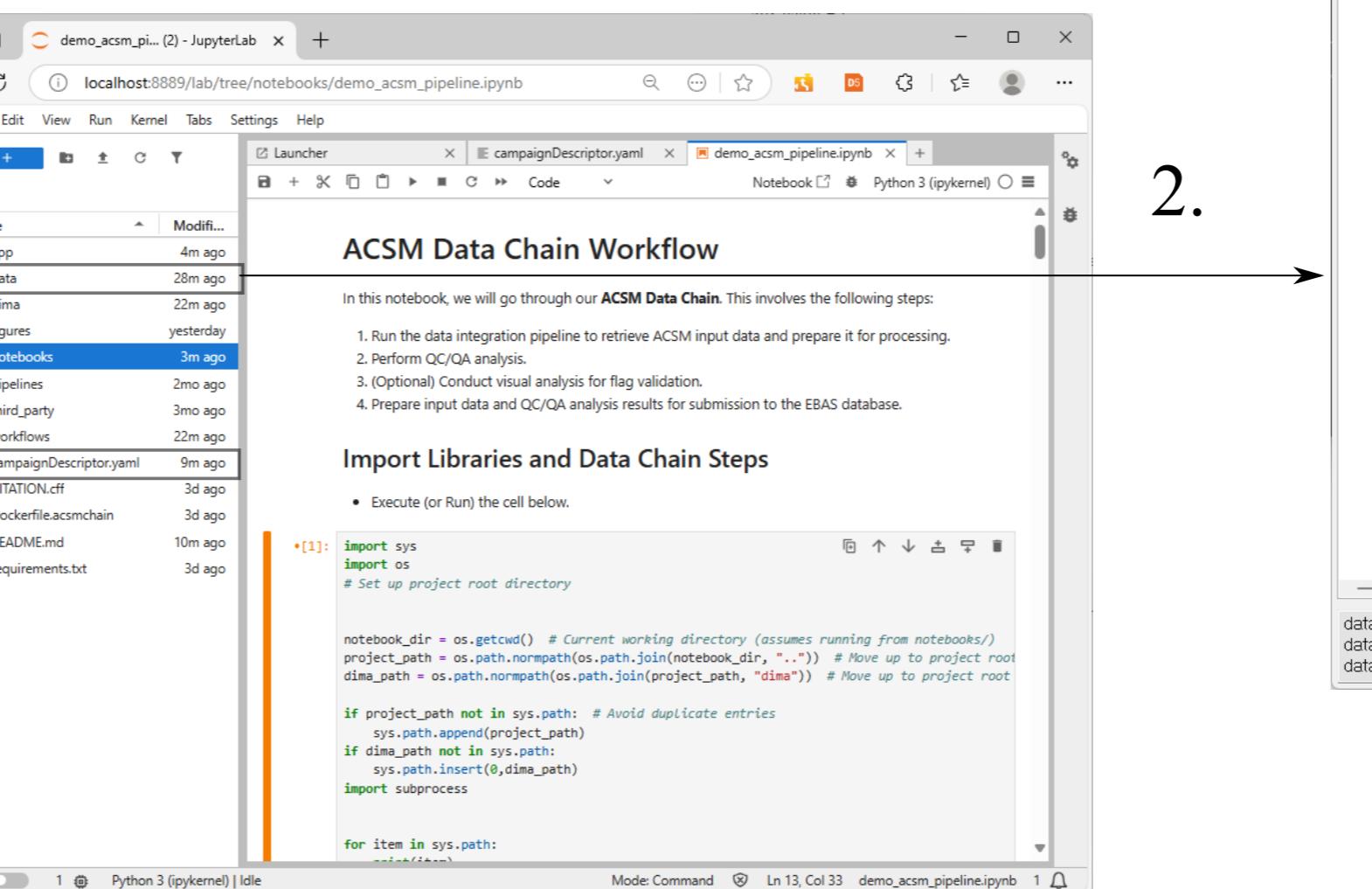


CampaignDescriptor.YAML

```
input_file_directory: '/mnt/network/lac_ord/Data/JFJ/'  
output_file_directory: '../data/'  
  
project: 'Building FAIR data chains for atmospheric observations in the ACTRIS Switzerland Network'  
contact: 'LeilaS'  
group_id: '5502' # APOG  
  
experiment: 'acsm_campaign' # beamtime, smog_chamber, lab_experiment  
dataset_startdate:  
dataset_enddate:  
actris_level: '1'  
  
# Define an (output) filename format of a resulting hdf5 file  
station_abbr : 'JFJ' # Set this value as either "JFJ" or "PAY"  
year : '2024'  
instrument_name : 'ACSM-017' #'ACSM-092'  
filename_format : "station_abbr, year"  
  
instrument_datafolder:  
- "ACSM_TOFWARE/2024"  
  
integration_mode: 'collection'  
# Specify datetimes (YYYY-MM-DD HH-MM-SS) at which experimental steps were created.  
datetime_steps: []  
  
# ACSM data chain  
originator:  
name: Simon, Leila  
email: leila.simon@psi.ch  
affiliation: Paul Scherrer Institute, PSI  
department: Laboratory of Atmospheric Chemistry  
address_line_1: ""  
address_line_2: ""  
postal_code: 5232  
city: Villigen PSI  
country: Switzerland
```

Main Data Chain Jupyter Notebook

1.



The JupyterLab interface shows the 'demo_acsm_pipeline.ipynb' notebook. The left sidebar displays a tree view of the project structure, including 'campaignDescriptor.yaml'. The main area shows the notebook content:

ACSM Data Chain Workflow

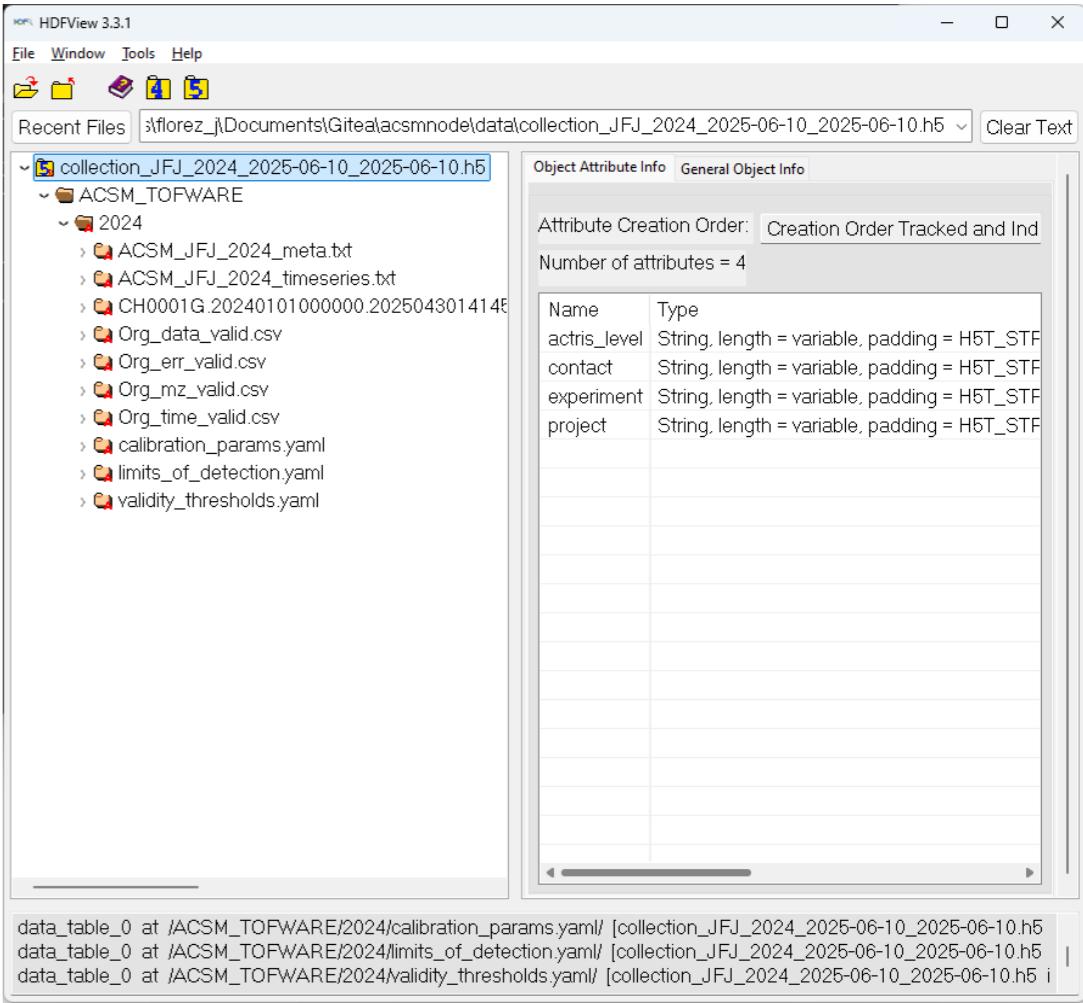
In this notebook, we will go through our **ACSM Data Chain**. This involves the following steps:

1. Run the data integration pipeline to retrieve ACSM input data and prepare it for processing.
2. Perform QC/QA analysis.
3. (Optional) Conduct visual analysis for flag validation.
4. Prepare input data and QC/QA analysis results for submission to the EBAS database.

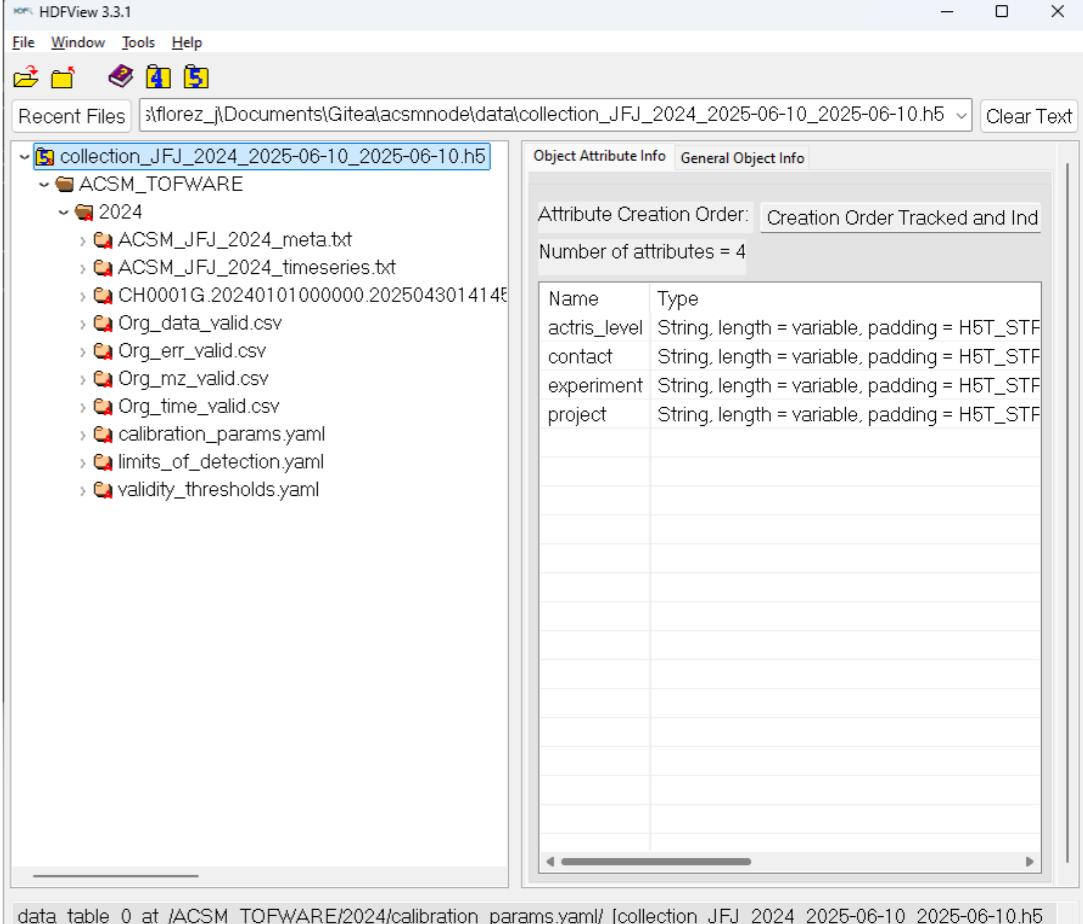
Import Libraries and Data Chain Steps

- Execute (or Run) the cell below.

```
[1]: import sys  
import os  
# Set up project root directory  
  
notebook_dir = os.getcwd() # Current working directory (assumes running from notebooks/)  
project_path = os.path.normpath(os.path.join(notebook_dir, "..")) # Move up to project root  
dima_path = os.path.normpath(os.path.join(project_path, "dima")) # Move up to project root  
  
if project_path not in sys.path: # Avoid duplicate entries  
    sys.path.append(project_path)  
if dima_path not in sys.path:  
    sys.path.insert(0, dima_path)  
import subprocess  
  
for item in sys.path:  
    print(item)
```

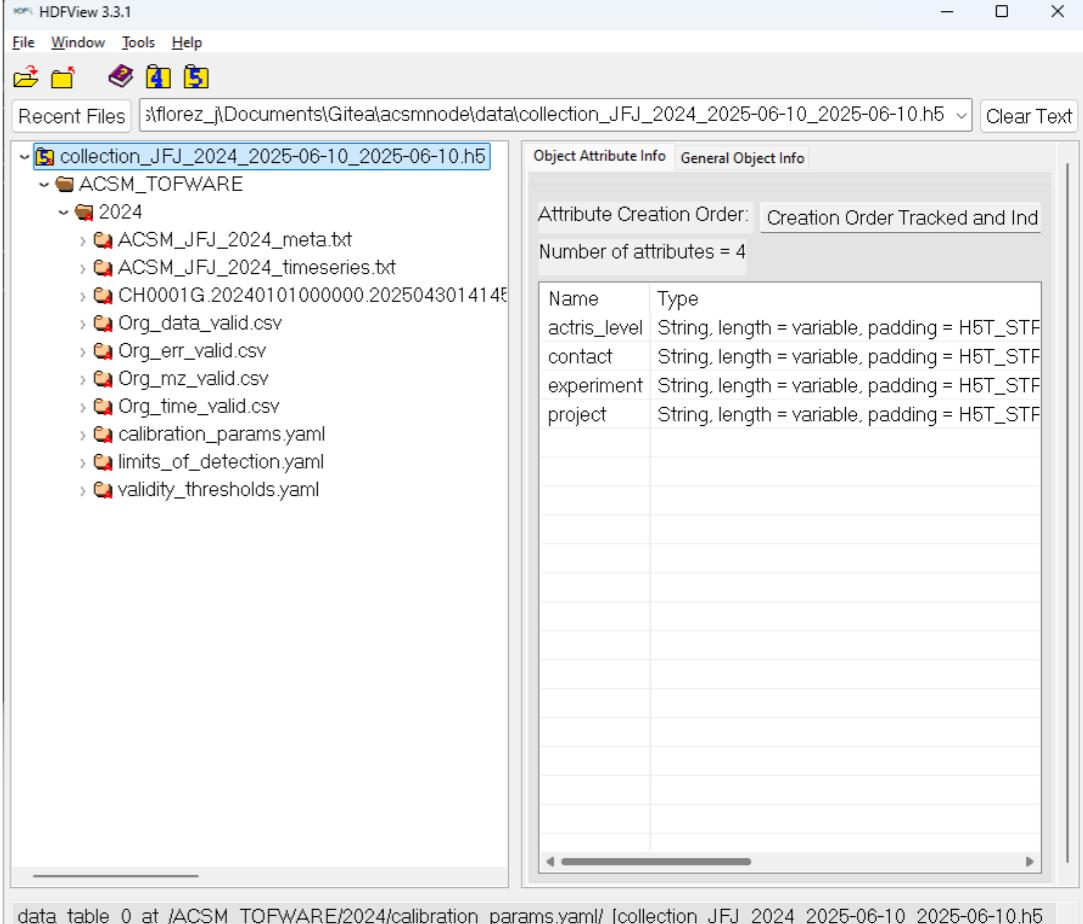


data_table_0 at /ACSM_TOFWARE/2024/calibration_params.yaml/ [collection_JFJ_2024_2025-06-10.h5]
data_table_0 at /ACSM_TOFWARE/2024/limits_of_detection.yaml/ [collection_JFJ_2024_2025-06-10.h5]
data_table_0 at /ACSM_TOFWARE/2024/validity_thresholds.yaml/ [collection_JFJ_2024_2025-06-10.h5]



data_table_0 at /ACSM_TOFWARE/2024/calibration_params.yaml/ [collection_JFJ_2024_2025-06-10.h5]
data_table_0 at /ACSM_TOFWARE/2024/limits_of_detection.yaml/ [collection_JFJ_2024_2025-06-10.h5]
data_table_0 at /ACSM_TOFWARE/2024/validity_thresholds.yaml/ [collection_JFJ_2024_2025-06-10.h5]

data_table_0 at /ACSM_TOFWARE/2024/calibration_params.yaml/ [collection_JFJ_2024_2025-06-10.h5]
data_table_0 at /ACSM_TOFWARE/2024/limits_of_detection.yaml/ [collection_JFJ_2024_2025-06-10.h5]
data_table_0 at /ACSM_TOFWARE/2024/validity_thresholds.yaml/ [collection_JFJ_2024_2025-06-10.h5]



data_table_0 at /ACSM_TOFWARE/2024/calibration_params.yaml/ [collection_JFJ_2024_2025-06-10.h5]
data_table_0 at /ACSM_TOFWARE/2024/limits_of_detection.yaml/ [collection_JFJ_2024_2025-06-10.h5]
data_table_0 at /ACSM_TOFWARE/2024/validity_thresholds.yaml/ [collection_JFJ_2024_2025-06-10.h5]