UN Data Hackathon

extracting AIS data from UNGP

Team : GEMy

group email :

gemy@dosm.gov.my (mailto:gemy@dosm.gov.my)

individual email:

rajkumar@dosm.gov.my (mailto:rajkumar@dosm.gov.my) ; shukor.talib@dosm.gov.my (mailto:shukor.talib@dosm.gov.my) ; nurhuda@dosm.gov.my (mailto:nurhuda@dosm.gov.my) ; tgnoradilah@dosm.gov.my (mailto:tgnoradilah@dosm.gov.my) ; najmi.ariffin@dosm.gov.my (mailto:najmi.ariffin@dosm.gov.my)

In [1]:

```
#allow multiple outputs in one jupyter cell
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

import pandas as pd
from datetime import datetime
# to apply aggregation functions on spark df
import pyspark.sql.functions as F
```

In [2]:

```
# this cell contains the code to access GitLab repo
# need it to install ais package from GitLab repo
import sys
import subprocess

GITLAB_USER = "read_aistt"  # read only access
GITLAB_TOKEN = "MMQ6ky1rnLsuKxjyZuvB"

# clone the repo and install the ais packag
git_package = f"git+https://{GITLAB_USER}:{GITLAB_TOKEN}@code.officialstatistics.org/trade-

std_out = subprocess.run([sys.executable, "-m", "pip", "install", git_package], capture_out
print(std_out)
```

```
Collecting git+https://read_aistt:****@code.officialstatistics.org/trade-tas
k-team-phase-1/ais.git
  Cloning https://read_aistt:****@code.officialstatistics.org/trade-task-tea
m-phase-1/ais.git (https://read_aistt:****@code.officialstatistics.org/trade
-task-team-phase-1/ais.git) to /tmp/pip-req-build-3p5s810y
Building wheels for collected packages: ais
  Building wheel for ais (setup.py): started
  Building wheel for ais (setup.py): finished with status 'done'
  Created wheel for ais: filename=ais-2.7.6-py3-none-any.whl size=9267 sha25
6=050e877075184ea051f78f61665e66484c2bf6c3165c83f0eaa71e002df07a0c
  Stored in directory: /tmp/pip-ephem-wheel-cache-fjpleh21/wheels/49/e0/a2/2
5d96a62cf626776ab2fd57fcbd822c2b8118049a84b16953d
Successfully built ais
Installing collected packages: ais
Successfully installed ais-2.7.6
```

In [3]:

```
# import get_ais() from ais package
from ais import functions as af
```

In [4]:

```
# details about the function e.g.
    # input parameters,
    # output: spark df
    # usage of this function in examples below
af.get_ais?
```

```
Signature:
af.get_ais(
    spark: pyspark.sql.session.SparkSession,
    start_date: datetime.datetime,
    end_date: datetime.datetime = None,
    h3_list: Union[List[int], NoneType] = None,
    polygon_hex_df: Union[pyspark.sql.dataframe.DataFrame, NoneType] = None,
    mmsi_list: Union[List[int], NoneType] = None,
    message_type: Union[List[int], NoneType] = [1, 2, 3, 4, 18, 19, 27],
    columns: Union[List[str], NoneType] = ['*'],
    polygon: Union[Dict, NoneType] = None,
    polygon_hex_resolution: Union[int, NoneType] = 8,
) -> pyspark.sql.dataframe.DataFrame
Docstring:
A wrapper function to apply filters on the AIS data.
Note that default parameters for message type are
position message types

Parameters
----------
spark: SparkSession

start_date: datetime
    the start date filter to apply

end_date: datetime
    the end date filter to apply. To filter a single date, use end_date equa
l to start_date

h3_list: list of int, default None
    h3 indices must be in int format and must have the same resolution.
    if None then it is not applied.

polygon_hex_df: dataframe, from polygon_to_hex_df function
    Dataframe with the following columns (minimum columns to contain) :
    - hex_id: the h3 hex ids (64-bit ints)
    - polygon_name: the name of the polygon
    - hex_resolution: the resolution of the hex (should be the same for all)
    The hex_ids should be contained in only one polygon_name, otherwise resu
lting dataframe
    will contain duplicate entries.

mmsi_list: list of int, default None
    the list of mmsi filter to apply. if None, then it is not applied

message_type: list of int, default [1,2,3,4,18,19,27] <- position messages
    the list of message types to retain. if not supplied then the default me
ssage type filter is applied
    use ["*"] to get all message types

columns: list of str, default ["*"]
    the list of columns to retain. if not supplied, all columns are returned
```

```
polygon: Optional[Dict] = None
    GeoJson representation of polygon. If supplied, then the hex approximati
on of the polygon
    is calculated using poly_container (polygon, hex_resolution, overfill=Tr
ue). The AIS data
    will be filtered according to hexes first  and then according to polygon
using
    Sedona functions:

    select *,  ST_Point(longitude,latitude) as point, ST_GeomFromGeoJSON('{p
olygon}') as
    polygon from temp where ST_Within(point, polygon)


polygon_hex_resolution: int = 8
    The resolution of the hexagons to fill the input polygon with. Default i
s 8, a hex with an avg area of 0.737 sq km.
    A polygon with an area of 100 sq. km will contain ~136 resolution 8 hexe
s. The same 100 sq. km polygon
    can be approximated by ~949 hexes using resolution 9. Note that the high
er the resolution, the higher
    the polygon area covered by the hexes. However, a small increase in reso
lution dramatically increases
    the number of hexes. See https://h3geo.org/docs/core-library/restable/
 (https://h3geo.org/docs/core-library/restable/) for a table of hex resoluti
ons.

Returns
-------
Spark dataframe with the filters applied.

Notes
-----
If multiple filters are provided, the most restrictive filters are applied.
 For example, both polygon and h3_list
are provided where h3_list is a list of hexes fully contained within the pol
ygon. The filtered AIS data will only contain
those within the hexes. Data within the polygon but outside the hexes will n
ot be included.
File:      /opt/conda/lib/python3.8/site-packages/ais/_aisfilter.py
Type:      function
```

In [5]:

```
start_date = datetime.fromisoformat("2022-01-01")

df = af.get_ais(spark, start_date)
df.show(n=1, vertical=True, truncate=False)
```

```
-RECORD 0-------------------------------------------------------------------
--------------------------------------------------------
 message_type       | 1
 mmsi               | 205654000
 imo                | 9691279
 vessel_name        | DN97
 callsign           | ORRK
 vessel_type        | Port Tender
 vessel_type_code   | 53
 vessel_type_cargo  | null
 vessel_class       | A
 length             | 17.0
 width              | 6.0
 flag_country       | Belgium
 flag_code          | 205
 destination        | ZEEBRUGGE
 eta                | 10081400
 draught            | 1.2
 longitude          | 3.20316
 latitude           | 51.32248833
 sog                | 0.0
 cog                | 0.0
 rot                | 0.0
 heading            | 0.0
 nav_status         | Not Defined
 nav_status_code    | 15
 source             | T-AIS
 dt_pos_utc         | 2022-01-01 21:17:43
 dt_static_utc      | 2022-01-01 21:14:24
 dt_insert_utc      | 2022-01-01 21:17:48
 vessel_type_main   | null
 vessel_type_sub    | null
 eeid               | 4897682788452534256
 source_filename    | s3a://ungp-ais-data-historical-backup/exact-earth-data/
nonprod/year=2022/month=01/day=01/20220101211833.csv.gz
 H3index_0          | 8019fffffffffff
 H3_int_index_0     | 576918149140578303
 H3_int_index_1     | 581412952674926591
 H3_int_index_2     | 585913253767413759
 H3_int_index_3     | 590416715955830783
 H3_int_index_4     | 594920306993266687
 H3_int_index_5     | 599423900178186239
 H3_int_index_6     | 603927499402903551
 H3_int_index_7     | 608431098929610751
 H3_int_index_8     | 612934698542301183
 H3_int_index_9     | 617438298168098815
 H3_int_index_10    | 621941897795403775
 H3_int_index_11    | 626445497422761983
 H3_int_index_12    | 630949097050129919
 H3_int_index_13    | 635452696677454463
 H3_int_index_14    | 639956296304824911
 H3_int_index_15    | 644459895932195403
only showing top 1 row
```

In [6]:

```
#malaysia
columns = ["mmsi", "latitude", "longitude", "dt_insert_utc", "eeid", "flag_country", "flag_
           "eta", "vessel_type_main", "imo", "vessel_type","vessel_type_code","vessel_type_
          "destination", "vessel_type_sub" ]

start_date = datetime.fromisoformat("2022-01-01")
end_date = datetime.fromisoformat("2022-01-31")
df = af.get_ais(spark,
                start_date,
                end_date = end_date,
                columns=columns)
df.count()
```

Out[6]:

```
708862027
```

In [7]:

```
# filter mmsi

mmsi_list = [533131111,
             533131162,
             311053500,
             357900000,
             370220000,
             209444000,
             235479000,
             249675000,
             255805778,
             309046000,
             311053500,
             312067000,
             370220000,
             374898000,
             477311600,
             565711000,
             413212060 ]

columns = ["mmsi", "latitude", "longitude", "dt_insert_utc", "eeid", "flag_country", "flag_
           "eta", "vessel_type_main", "imo", "vessel_type","vessel_type_code","vessel_type_
          "destination", "vessel_type_sub" ]

start_date = datetime.fromisoformat("2022-01-01")
end_date = datetime.fromisoformat("2022-01-31")

df = af.get_ais(spark,
                start_date,
                end_date = end_date,
                mmsi_list = mmsi_list,
                columns = columns)

df.count()
```

Out[7]:

```
93140
```

In [8]:

```
df.show(n=1, vertical=True, truncate=False)
```

```
-RECORD 0-------------------------------------------------------------------
--
 mmsi              | 235479000
 latitude          | 5.86358667
 longitude         | 96.49673167
 dt_insert_utc     | 2022-01-01 22:51:48
 eeid              | 5115565929652625368
 flag_country      | UK
 flag_code         | 235
 eta               | 1080600
 vessel_type_main  | Container Ship
 imo               | 9241322
 vessel_type       | Cargo
 vessel_type_code  | 74
 vessel_type_cargo | Carrying DG,HS or MP,IMO hazard or Pollutant Category O
S
 destination       | MYPKG=>AEJBA
 vessel_type_sub   | null
only showing top 1 row
```

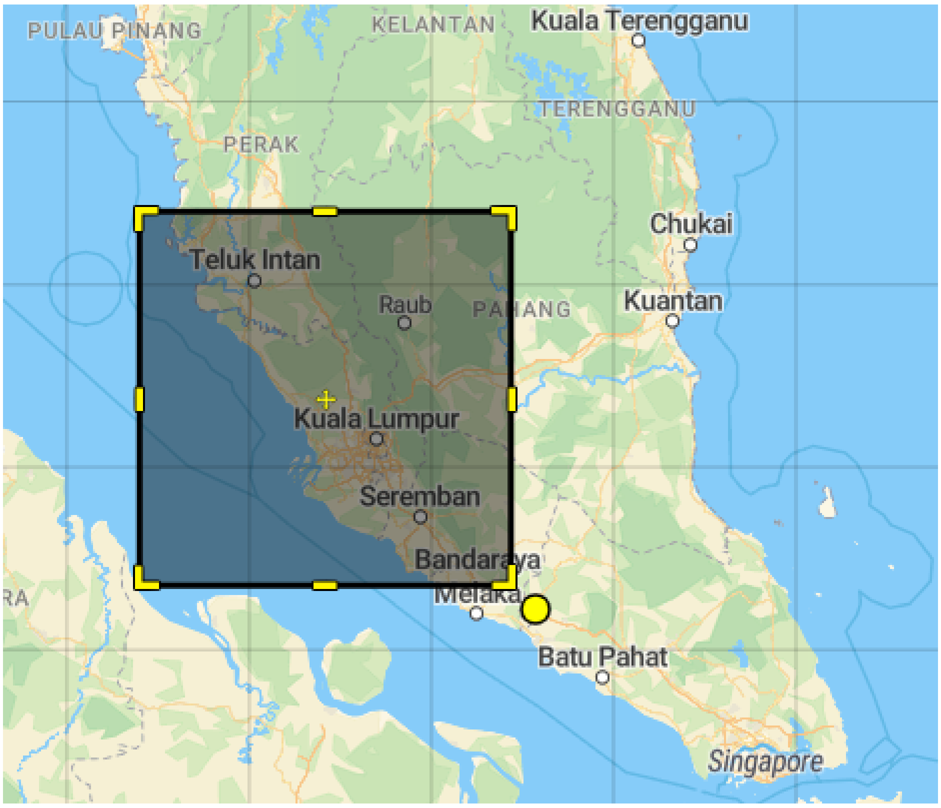
In [9]:

```
# first this function and then pass on its output with get_ais()
af.polygon_to_hex_df?
```

```
Signature:
af.polygon_to_hex_df(
    polygons: List[Tuple[str, Dict]],
    hex_resolution: int = 8,
    overfill=False,
) -> pandas.core.frame.DataFrame
Docstring:
A wrapper for h3.polyfill that returns integer hex ids for multiple polygo
ns.

Parameters
----------
polygons: list of tuples
    the first element in this tuple is expected to be a (name) string
    identifier for the polygon and the second element is the polygon itsel
f (see example above)

hex_resolution: int, default 8
    the resolution of the hexagons to fill the input polygon with. Default
is 8, a hex with an avg area of 0.737 sq km.
    A polygon with an area of 100 sq. km will contain ~136 resolution 8 he
xes. The same 100 sq. km polygon
    can be approximated by ~949 hexes using resolution 9. Note that the hi
gher the resolution, the higher
    the polygon area covered by the hexes. However, a small increase in re
solution dramatically increases
    the number of hexes. See https://h3geo.org/docs/core-library/restable/
(https://h3geo.org/docs/core-library/restable/) for a table of hex resolut
ions.

Returns
-------
Dataframe with the following columns:
    - hex_id: the h3 hex ids (64-bit ints)
    - polygon_name: the name of the polygon
    - hex_resolution: the resolution of the hex

>>>hull_bbox = {
    "type": "Polygon",
    "coordinates": [
        [
            [-0.3169, 53.7344],
            [-0.2537, 53.7344],
            [-0.2537, 53.75],
            [-0.3169, 53.75],
            [-0.3169, 53.7344]
        ]
    ]
}
>>>london_bbox = {
    "type": "Polygon",
    "coordinates": [
        [
            [-0.1203, 51.4415],
            [0.5869, 51.4415],
            [0.5869, 51.5262],
            [-0.1203, 51.5262],
            [-0.1203, 51.4415]
        ]
    ]
}
>>>query_polys = [("HullPortArea", hull_bbox),("LondonPortArea", london_bb
ox)]
>>>polygon_to_hex_df(query_polys, 10)
                  hex_id    polygon_name  hex_resolution
0      621940969126789119    HullPortArea              10
1      621940969212772351    HullPortArea              10
2      621940969214869503    HullPortArea              10
3      621940974334017535    HullPortArea              10
4      621940969137274879    HullPortArea              10
                     ...             ...             ...
34427  621941942979756031  LondonPortArea              10
34428  621941942943285247  LondonPortArea              10
34429  621941942854713343  LondonPortArea              10
34430  621941942818242559  LondonPortArea              10
34431  621941940535295999  LondonPortArea              10
File:      /opt/conda/lib/python3.8/site-packages/ais/_poly.py
Type:      function
```

In [10]:

```
# 2nd parameter for polygon_to_hex_df()
    #   https://boundingbox.klokantech.com/
# polygon coordinates in geojson format
malaysia_polygon = {
        "type": "Polygon",
        "coordinates": [
            [
                [101.1134319752,2.8025194725],
                [101.4495449513,2.8025194725],
                [101.4495449513,3.1131551166],
                [101.1134319752,3.1131551166],
                [101.1134319752,2.8025194725]
            ]
        ]
    }
```



In [11]:

```
# first parameter for polygon_to_hex_df() is the name/label for the polygon
polygon_hex_df_malaysia = af.polygon_to_hex_df([("Malaysia_Port_Polygon", malaysia_polygon)
```

In [12]:

```
start_date = datetime.fromisoformat("2022-01-01")
end_date = datetime.fromisoformat("2022-01-31")
columns = ["mmsi", "latitude", "longitude", "dt_insert_utc", "eeid", "flag_country", "flag_
           "eta", "vessel_type_main", "imo", "vessel_type","vessel_type_code","vessel_type_
          "destination", "vessel_type_sub" ]

# pass polygon_hex_df to get_ais()
df = af.get_ais(spark,
                start_date,
                end_date = end_date,
                columns = columns,
                polygon_hex_df = polygon_hex_df_malaysia

               )

df.count()
```

Out[12]:

```
797879
```

In [13]:

```
mmsi_list = [533131111,
             533131162,
             311053500,
             357900000,
             370220000,
             209444000,
             235479000,
             249675000,
             255805778,
             309046000,
             311053500,
             312067000,
             370220000,
             374898000,
             477311600,
             565711000,
             413212060 ]

start_date = datetime.fromisoformat("2022-01-01")
end_date = datetime.fromisoformat("2022-01-31")
columns = ["mmsi", "latitude", "longitude", "dt_insert_utc", "eeid",
           "flag_country", "flag_code",
           "eta", "vessel_type_main", "imo", "vessel_type","vessel_type_code","vessel_type_
          "destination", "vessel_type_sub" ]

# pass polygon_hex_df to get_ais()
df = af.get_ais(spark,
                start_date,
                end_date = end_date,
                columns = columns,
                mmsi_list = mmsi_list,
                polygon_hex_df = polygon_hex_df_malaysia

               )

df.count()
```

Out[13]:

```
20701
```

In [14]:

```
# ais messages captured in the Malaysia port region
df.show(n=1)
```

```
+--------------+------------+---------+------------+-------+-----------+----
------------+-----------------+-----------+-------------------+---------+---
----+------------------+----------+----------------+-------------------+----
-----------+--------------------+
|hex_resolution|   longitude|     mmsi|flag_country|    eta|destination|vess
el_type_code|vessel_type_cargo|vessel_type|               eeid|flag_code|
imo|    H3_int_index_8|  latitude|vessel_type_main|      dt_insert_utc|vesse
l_type_sub|        polygon_name|
+--------------+------------+---------+------------+-------+-----------+----
------------+-----------------+-----------+-------------------+---------+---
----+------------------+----------+----------------+-------------------+----
-----------+--------------------+
|             8|101.30643167|209444000|      Cyprus|1011600|      MYPKG|
70|             null|      Cargo|4705970841525673120|      209|9507714|61426
6716666462207|2.80780333|            null|2022-01-01 17:42:21|           nul
l|Malaysia_Port_Pol...|
+--------------+------------+---------+------------+-------+-----------+----
------------+-----------------+-----------+-------------------+---------+---
----+------------------+----------+----------------+-------------------+----
-----------+--------------------+
only showing top 1 row
```

In [15]:

```
pd_df = df.toPandas()

type(pd_df)
pd_df.shape
```

Out[15]:

```
pandas.core.frame.DataFrame
```

Out[15]:

```
(20701, 18)
```

In [16]:

```
pd_df.head()
```

Out[16]:

	hex_resolution	longitude	mmsi	flag_country	eta	destination	vessel_type_code
0	8	101.306432	209444000	Cyprus	1011600	MYPKG	70
1	8	101.309188	209444000	Cyprus	1011600	MYPKG	70
2	8	101.306525	209444000	Cyprus	1011600	MYPKG	70
3	8	101.306937	209444000	Cyprus	1011600	MYPKG	70
4	8	101.306852	209444000	Cyprus	1011600	MYPKG	70

In [17]:

```
!pip install s3fs
import s3fs

# create a handle for s3fs
fs = s3fs.S3FileSystem(anon=False)
```

```
WARNING: The directory '/home/sparkuser/.cache/pip' or its parent directory
 is not owned or is not writable by the current user. The cache has been dis
abled. Check the permissions and owner of that directory. If executing pip w
ith sudo, you should use sudo's -H flag.
Collecting s3fs
  Downloading s3fs-2022.11.0-py3-none-any.whl (27 kB)
Collecting fsspec==2022.11.0
  Downloading fsspec-2022.11.0-py3-none-any.whl (139 kB)
     |████████████████████████████████| 139 kB 52.3 MB/s eta 0:00:01
Collecting aiohttp!=4.0.0a0,!=4.0.0a1
  Downloading aiohttp-3.8.3-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x8
6_64.whl (1.0 MB)
     |████████████████████████████████| 1.0 MB 128.7 MB/s eta 0:00:01
Collecting aiobotocore~=2.4.0
  Downloading aiobotocore-2.4.0-py3-none-any.whl (65 kB)
     |████████████████████████████████| 65 kB 98.6 MB/s  eta 0:00:01
Collecting botocore<1.27.60,>=1.27.59
  Downloading botocore-1.27.59-py3-none-any.whl (9.1 MB)
     |████████████████████████████████| 9.1 MB 99.7 MB/s eta 0:00:01
Collecting aioitertools>=0.5.1
  Downloading aioitertools-0.11.0-py3-none-any.whl (23 kB)
Collecting wrapt>=1.10.10
  Downloading wrapt-1.14.1-cp38-cp38-manylinux_2_5_x86_64.manylinux1_x86_64.
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (81 kB)
     |████████████████████████████████| 81 kB 95.3 MB/s eta 0:00:01
Collecting yarl<2.0,>=1.0
  Downloading yarl-1.8.1-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_6
4.whl (262 kB)
     |████████████████████████████████| 262 kB 129.8 MB/s eta 0:00:01
Collecting async-timeout<5.0,>=4.0.0a3
  Downloading async_timeout-4.0.2-py3-none-any.whl (5.8 kB)
Collecting multidict<7.0,>=4.5
  Downloading multidict-6.0.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_
x86_64.whl (121 kB)
     |████████████████████████████████| 121 kB 125.9 MB/s eta 0:00:01
Collecting frozenlist>=1.1.1
  Downloading frozenlist-1.3.3-cp38-cp38-manylinux_2_5_x86_64.manylinux1_x86
_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (161 kB)
     |████████████████████████████████| 161 kB 118.3 MB/s eta 0:00:01
Collecting charset-normalizer<3.0,>=2.0
  Downloading charset_normalizer-2.1.1-py3-none-any.whl (39 kB)
Requirement already satisfied: attrs>=17.3.0 in /opt/conda/lib/python3.8/sit
e-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->s3fs) (21.2.0)
Collecting aiosignal>=1.1.2
  Downloading aiosignal-1.3.1-py3-none-any.whl (7.6 kB)
Collecting typing_extensions>=4.0
  Downloading typing_extensions-4.4.0-py3-none-any.whl (26 kB)
Collecting jmespath<2.0.0,>=0.7.1
  Downloading jmespath-1.0.1-py3-none-any.whl (20 kB)
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /opt/conda/lib/pytho
n3.8/site-packages (from botocore<1.27.60,>=1.27.59->aiobotocore~=2.4.0->s3f
s) (1.26.4)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /opt/conda/li
```

```
b/python3.8/site-packages (from botocore<1.27.60,>=1.27.59->aiobotocore~=2.
4.0->s3fs) (2.8.1)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.8/site-pac
kages (from python-dateutil<3.0.0,>=2.1->botocore<1.27.60,>=1.27.59->aioboto
core~=2.4.0->s3fs) (1.15.0)
Requirement already satisfied: idna>=2.0 in /opt/conda/lib/python3.8/site-pa
ckages (from yarl<2.0,>=1.0->aiohttp!=4.0.0a0,!=4.0.0a1->s3fs) (2.10)
Installing collected packages: multidict, frozenlist, yarl, typing-extension
s, jmespath, charset-normalizer, async-timeout, aiosignal, wrapt, botocore,
 aioitertools, aiohttp, fsspec, aiobotocore, s3fs
Successfully installed aiobotocore-2.4.0 aiohttp-3.8.3 aioitertools-0.11.0 a
iosignal-1.3.1 async-timeout-4.0.2 botocore-1.27.59 charset-normalizer-2.1.1
frozenlist-1.3.3 fsspec-2022.11.0 jmespath-1.0.1 multidict-6.0.2 s3fs-2022.1
1.0 typing-extensions-4.4.0 wrapt-1.14.1 yarl-1.8.1
WARNING: Running pip as root will break packages and permissions. You should
install packages reliably by using venv: https://pip.pypa.io/warnings/venv
 (https://pip.pypa.io/warnings/venv)
WARNING: You are using pip version 21.1.2; however, version 22.3.1 is availa
ble.
You should consider upgrading via the '/opt/conda/bin/python -m pip install
 --upgrade pip' command.
```

In [19]:

```
af.create_download_link(pd_df[:1000], title = "Download CSV file", filename = "myresults_mm
```

Out[19]:

Download CSV file

In [20]:

```
af.create_download_link(pd_df[1000:5000], title = "Download CSV file", filename = "myresult
```

Out[20]:

Download CSV file

In [ ]:

```
af.create_download_link(pd_df[5000:10000], title = "Download CSV file", filename = "myresul
```

In [ ]:

```
#af.create_download_link(pd_df[10000:15000], title = "Download CSV file", filename = "myres
```

In [ ]:

```
#af.create_download_link(pd_df[15000:20000], title = "Download CSV file", filename = "myres
```

In [ ]:

```
!free -m
```