## Spark notebook

This notebook will only work in a Jupyter notebook or Jupyter lab session running on the cluster master node in the cloud.

Follow the instructions on the computing resources page to start a cluster and open this notebook.

**Steps**

1. Connect to the Windows server using Windows App.
2. Connect to Kubernetes.
3. Start Jupyter and open this notebook from Jupyter in order to connect to Spark.

```
# Run this cell to import pyspark and to define start_spark() and stop_spark()

import findspark

findspark.init()

import getpass
import random
import re

import pandas
import pyspark
from IPython.display import HTML, display
from pyspark import SparkContext
from pyspark.sql import SparkSession

# Constants used to interact with Azure Blob Storage using the hdfs command or Spark

global username

username = re.sub("@.*", "", getpass.getuser())

global azure_account_name
global azure_data_container_name
global azure_user_container_name
global azure_user_token

azure_account_name = "madsstorage002"
azure_data_container_name = "campus-data"
azure_user_container_name = "campus-user"
azure_user_token = r"sp=racwdl&st=2025-08-01T09:41:33Z&se=2026-12-30T16:56:33Z&spr=https&sv=2024-11-04&sr=c&sig=Gz


# Functions used below


def dict_to_html(d):
    """Convert a Python dictionary into a two column table for display."""

    html = []

    html.append(f'<table width="100%" style="width:100%; font-family: monospace;">')
    for k, v in d.items():
        html.append(f'<tr><td style="text-align:left;">{k}</td><td>{v}</td></tr>')
    html.append(f"</table>")

    return "".join(html)


def show_as_html(df, n=20):
    """Leverage existing pandas jupyter integration to show a spark dataframe as html.

    Args:
        n (int): number of rows to show (default: 20)
    """

    display(df.limit(n).toPandas())


def display_spark():
    """Display the status of the active Spark session if one is currently running."""

    if "spark" in globals() and "sc" in globals():

        name = sc.getConf().get("spark.app.name")

        html = [
            f"<p><b>Spark</b></p>",
            f'<p>The spark session is <b><span style="color:green">active</span></b>, look for <code>{name}</code>
```

```
            f"<ul>",
            f'<li><a href="http://localhost:{sc.uiWebUrl.split(":")[-1]}" target="_blank">Spark Application UI</a>
            f"</ul>",
            f"<p><b>Config</b></p>",
            dict_to_html(dict(sc.getConf().getAll())),
            f"<p><b>Notes</b></p>",
            f"<ul>",
            f"<li>The spark session <code>spark</code> and spark context <code>sc</code> global variables have bee
            f"<li>Please run <code>stop_spark()</code> before closing the notebook or restarting the kernel or kil
            f"</ul>",
        ]
        display(HTML("".join(html)))

    else:

        html = [
            f"<p><b>Spark</b></p>",
            f'<p>The spark session is <b><span style="color:red">stopped</span></b>, confirm that <code>{username}
            f"<ul>",
            f'<li><a href="http://mathmadslinux2p.canterbury.ac.nz:8080/" target="_blank">Spark UI</a></li>',
            f"</ul>",
        ]
        display(HTML("".join(html)))


# Functions to start and stop spark


def start_spark(
    executor_instances=2, executor_cores=1, worker_memory=1, master_memory=1
):
    """Start a new Spark session and define globals for SparkSession (spark) and SparkContext (sc).

    Args:
        executor_instances (int): number of executors (default: 2)
        executor_cores (int): number of cores per executor (default: 1)
        worker_memory (float): worker memory (default: 1)
        master_memory (float): master memory (default: 1)
    """

    global spark
    global sc

    cores = executor_instances * executor_cores
    partitions = cores * 4
    port = 4000 + random.randint(1, 999)

    spark = (
        SparkSession.builder.config(
            "spark.driver.extraJavaOptions",
            f"-Dderby.system.home=/tmp/{username}/spark/",
        )
        .config("spark.dynamicAllocation.enabled", "false")
        .config("spark.executor.instances", str(executor_instances))
        .config("spark.executor.cores", str(executor_cores))
        .config("spark.cores.max", str(cores))
        .config("spark.driver.memory", f"{master_memory}g")
        .config("spark.executor.memory", f"{worker_memory}g")
        .config("spark.driver.maxResultSize", "0")
        .config("spark.sql.shuffle.partitions", str(partitions))
        .config(
            "spark.kubernetes.container.image",
            "madsregistry001.azurecr.io/hadoop-spark:v3.3.5-openjdk-8",
        )
        .config("spark.kubernetes.container.image.pullPolicy", "IfNotPresent")
        .config("spark.kubernetes.memoryOverheadFactor", "0.3")
        .config("spark.memory.fraction", "0.1")
        .config(
            f"fs.azure.sas.{azure_user_container_name}.{azure_account_name}.blob.core.windows.net",
            azure_user_token,
        )
        .config("spark.app.name", f"{username} (notebook)")
        .getOrCreate()
    )
    sc = SparkContext.getOrCreate()

    display_spark()


def stop_spark():
    """Stop the active Spark session and delete globals for SparkSession (spark) and SparkContext (sc)."""

    global spark
```

```
    global sc

    if "spark" in globals() and "sc" in globals():

        spark.stop()

        del spark
        del sc

    display_spark()


# Make css changes to improve spark output readability

html = [
    "<style>",
    "pre { white-space: pre !important; }",
    "table.dataframe td { white-space: nowrap !important; }",
    "table.dataframe thead th:first-child, table.dataframe tbody th { display: none; }",
    "</style>",
]
display(HTML("".join(html)))
```

```
# Run this cell to start a spark session in this notebook

start_spark(executor_instances=4, executor_cores=2, worker_memory=4, master_memory=4)
```

```
Warning: Ignoring non-Spark config property: fs.azure.sas.campus-user.madsstorage002.blob.core.windows.net
Warning: Ignoring non-Spark config property: SPARK_DRIVER_BIND_ADDRESS
25/09/11 17:59:21 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
```

### Spark

The spark session is **active**, look for `yxi75 (notebook)` under the running applications section in the Spark UI.

- Spark Application UI

### Config

| | |
|---|---|
| spark.dynamicAllocation.enabled | false |
| spark.fs.azure.sas.uco-user.madsstorage002.blob.core.windows.net | "sp=racwdl&st=2024-09-19T08:00:18Z&se=2025-09-19T16:00:18Z&spr=https&sv=2022-11-02&sr=c&sig=qtg6fCdoFz6k3EJLw7dA8D3D8wN0neAYw8yG4z4Lw2o%3D" |
| spark.kubernetes.driver.pod.name | spark-master-driver |
| spark.executor.instances | 4 |
| spark.driver.memory | 4g |
| spark.app.name | yxi75 (notebook) |
| spark.fs.azure.sas.campus-user.madsstorage002.blob.core.windows.net | "sp=racwdl&st=2024-09-19T08:03:31Z&se=2025-09-19T16:03:31Z&spr=https&sv=2022-11-02&sr=c&sig=kMP%2BsBsRzdVVR8rrg%2BNbDhkRBNs6Q98kYY695XMRFDU%3D" |
| spark.kubernetes.container.image.pullPolicy | IfNotPresent |
| spark.sql.shuffle.partitions | 32 |
| spark.kubernetes.namespace | yxi75 |
| spark.serializer.objectStreamReset | 100 |
| spark.driver.maxResultSize | 0 |
| spark.app.startTime | 1757570361973 |
| spark.submit.deployMode | client |
| spark.master | k8s://https://kubernetes.default.svc.cluster.local:443 |
| spark.app.id | spark-7adcb5083a5b42329c59ab38a40282f7 |
| | -Djava.net.preferIPv6Addresses=false -XX:+IgnoreUnrecognizedVMOptions --add-opens=java.base/java.lang=ALL-UNNAMED --add- |

| | |
|---|---|
| spark.driver.extraJavaOptions | -Dderby.system.home=/tmp/yxi75/spark/ -Djdk.reflect.useDirectMethodHandle=false --add-opens=java.security.jgss/sun.security.krb5=ALL-UNNAMED --add-opens=java.base/sun.util.calendar=ALL-UNNAMED --add-opens=java.base/sun.security.action=ALL-UNNAMED --add-opens=java.base/sun.nio.cs=ALL-UNNAMED --add-opens=java.base/sun.nio.ch=ALL-UNNAMED --add-opens=java.base/jdk.internal.ref=ALL-UNNAMED --add-opens=java.base/java.util.concurrent.atomic=ALL-UNNAMED --add-opens=java.base/java.util.concurrent=ALL-UNNAMED --add-opens=java.base/java.util=ALL-UNNAMED --add-opens=java.base/java.nio=ALL-UNNAMED --add-opens=java.base/java.net=ALL-UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED --add-opens=java.base/java.lang.reflect=ALL-UNNAMED --add-opens=java.base/java.lang.invoke=ALL-UNNAMED --add-opens=java.base/java.lang=ALL-UNNAMED --add- |
| spark.fs.azure | org.apache.hadoop.fs.azure.NativeAzureFileSystem |
| spark.memory.fraction | 0.1 |
| spark.executor.memory | 4g |
| spark.executor.id | driver |
| spark.kubernetes.executor.container.image | madsregistry001.azurecr.io/hadoop-spark:v3.3.5-openjdk-8-1.0.16 |
| spark.executor.cores | 2 |
| spark.app.submitTime | 1757570361858 |
| spark.kubernetes.memoryOverheadFactor | 0.3 |
| spark.driver.host | spark-master-svc |
| spark.kubernetes.executor.podNamePrefix | yxi75-notebook-78045b99375b6dae |
| spark.ui.port | ${env:SPARK_UI_PORT} |
| spark.kubernetes.container.image | madsregistry001.azurecr.io/hadoop-spark:v3.3.5-openjdk-8 |
| spark.kubernetes.executor.podTemplateFile | /opt/spark/conf/executor-pod-template.yaml |
| fs.azure.sas.campus-user.madsstorage002.blob.core.windows.net | sp=racwdl&st=2025-08-01T09:41:33Z&se=2026-12-30T16:56:33Z&spr=https&sv=2024-11-04&sr=c&sig=GzR1hq7EJ0lRHj92oDO1MBNjkc602nrpfB5H8Cl7FFY%3D |
| spark.rdd.compress | True |
| spark.executor.extraJavaOptions | -Djava.net.preferIPv6Addresses=false -XX:+IgnoreUnrecognizedVMOptions --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.lang.invoke=ALL-UNNAMED --add-opens=java.base/java.lang.reflect=ALL-UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED --add-opens=java.base/java.net=ALL-UNNAMED --add-opens=java.base/java.nio=ALL-UNNAMED --add-opens=java.base/java.util=ALL-UNNAMED --add-opens=java.base/java.util.concurrent=ALL-UNNAMED --add-opens=java.base/java.util.concurrent.atomic=ALL-UNNAMED --add-opens=java.base/jdk.internal.ref=ALL-UNNAMED --add-opens=java.base/sun.nio.ch=ALL-UNNAMED --add-opens=java.base/sun.nio.cs=ALL-UNNAMED --add-opens=java.base/sun.security.action=ALL-UNNAMED --add-opens=java.base/sun.util.calendar=ALL-UNNAMED --add-opens=java.security.jgss/sun.security.krb5=ALL-UNNAMED -Djdk.reflect.useDirectMethodHandle=false |
| spark.cores.max | 8 |
| spark.driver.port | 7077 |
| spark.submit.pyFiles | |
| spark.ui.showConsoleProgress | true |

**Notes**

- The spark session `spark` and spark context `sc` global variables have been defined by `start_spark()`.
- Please run `stop_spark()` before closing the notebook or restarting the kernel or kill `yxi75 (notebook)` by hand using the link in the Spark UI.

```
# Write your imports here or insert cells below

import re
import subprocess
from math import asin, cos, radians, sin, sqrt
from pprint import pprint

import numpy as np
import pandas as pd
from pyspark.sql import functions as F
from pyspark.sql.functions import udf
from pyspark.sql.types import *
```

```
# Paths global variables
DATA_ROOT = "wasbs://campus-data@madsstorage002.blob.core.windows.net/ghcnd/"
USER_ROOT = "wasbs://campus-user@madsstorage002.blob.core.windows.net/yxi75/"

paths = {
    "daily": DATA_ROOT + "daily/",
    "stations": DATA_ROOT + "ghcnd-stations.txt",
    "countries": DATA_ROOT + "ghcnd-countries.txt",
    "states": DATA_ROOT + "ghcnd-states.txt",
    "inventory": DATA_ROOT + "ghcnd-inventory.txt",
}

paths

stations_enriched_savepath = USER_ROOT + "stations_enriched_parquet"
```

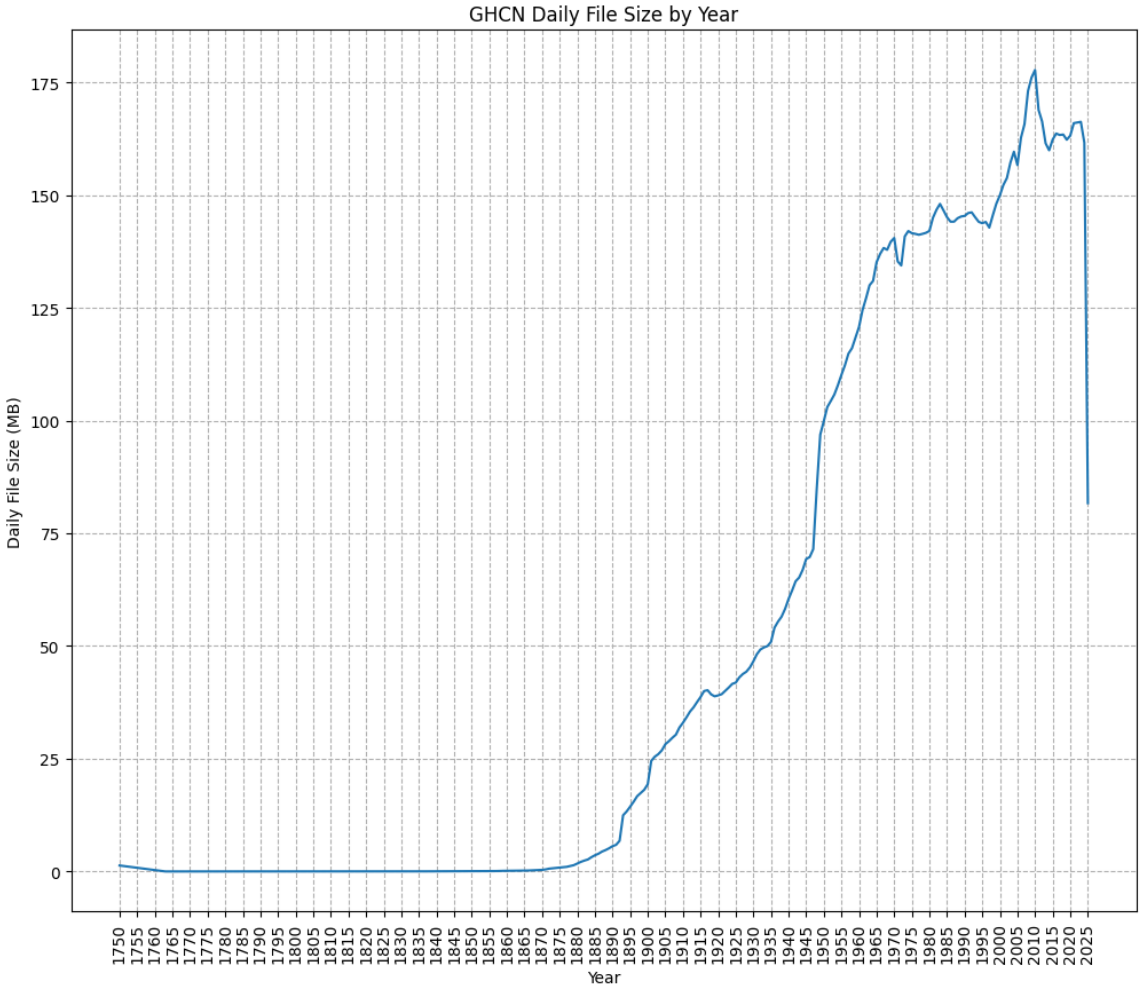
```
# Use the hdfs command to explore the data in Azure Blob Storage
# hdfs dfs-ls -h :Format file sizes in a human-readable fashion (eg 64.0m instead of 67108864).
# reference: https://hadoop.apache.org/docs/r3.3.5/hadoop-project-dist/hadoop-common/FileSystemShell.html#ls


# show DATA_ROOT Structure
!hdfs dfs -ls -h {DATA_ROOT}
```

```
Found 5 items
drwxrwxrwx   -          0 1970-01-01 12:00 wasbs://campus-data@madsstorage002.blob.core.windows.net/ghcnd/daily
-rwxrwxrwx   1      3.6 K 2025-08-01 21:31 wasbs://campus-data@madsstorage002.blob.core.windows.net/ghcnd/ghcnd-count
-rwxrwxrwx   1     33.6 M 2025-08-01 21:31 wasbs://campus-data@madsstorage002.blob.core.windows.net/ghcnd/ghcnd-inver
-rwxrwxrwx   1      1.1 K 2025-08-01 21:31 wasbs://campus-data@madsstorage002.blob.core.windows.net/ghcnd/ghcnd-state
-rwxrwxrwx   1     10.6 M 2025-08-01 21:31 wasbs://campus-data@madsstorage002.blob.core.windows.net/ghcnd/ghcnd-stati
```

```
# show Daily folder structure and export to txt file
!hdfs dfs -ls {paths["daily"]} > ./supplementary/daily_folder_index.txt
```

```
# load daily folder index info and plot to show data scale trend
daily_folder_index_path = "./supplementary/daily_folder_index.txt"
with open(file=daily_folder_index_path, mode="r") as f:
    lines = f.readlines()

data = []

for line in lines:
    # Skip "Found" line and directory entries
    if line.startswith("Found") or line.startswith("d"):
        continue

    # Unpack line into variables
    file_permissions, replication, size_bytes, mod_date, mod_time, path = line.split()

    # Append row dictionary
    data.append(
        {
            "file_permissions": file_permissions,
            "replication": int(replication),
            "size_bytes": int(size_bytes),
            "mod_date": mod_date,
            "mod_time": mod_time,
            "path": path,
        }
    )

# Convert list of dicts to DataFrame
```

```
df = pd.DataFrame(data)

# Add size in MB and extract year
df["size_MB"] = df["size_bytes"] / (1024 * 1024)
df["year"] = df.path.str.split("/").str[5].str.split(".").str[0].astype(int)

import matplotlib.pyplot as plt

plt.figure(figsize=(12, 10))
plt.plot(df["year"], df["size_MB"])
plt.xticks(range(df["year"].min(), df["year"].max() + 1, 5))
plt.xticks(rotation=90)
plt.xlabel("Year")
ymin = df["size_MB"].min()
ymax = df["size_MB"].max()
plt.ylabel("Daily File Size (MB)")
plt.title("GHCN Daily File Size by Year")
plt.grid(visible=True, axis="both", linestyle="--")
plt.savefig("./supplementary/GHCN Daily File Size by Year.png", dpi=300)
plt.show()
```




# Processing

## Q1 Data loading and simple EDA

First you will investigate the daily, stations, states, countries, and inventory data provided in cloud storage in wasbs://campus-data@madsstorage002.blob.core.windows.net/ghcnd/
using the hdfs command.
Follow the instructions in the notebook provided to explore each dataset using the hdfs command without loading any data into memory and answer the following questions:

### (a) How is the data structured? Are any of the datasets compressed?

Found 5 items under the directory 'wasbs://campus-data@madsstorage002.blob.core.windows.net/ghcnd/', including 1 directory and 4 text files as shown below:

- daily (264 csv.gz compressed csv files)

-- 1750.csv.gz
-- 1751.csv.gz
-- ...
-- 2025.csv.gz

- ghcnd-countries.txt
- ghcnd-inventory.txt
- ghcnd-states.txt
- ghcnd-stations.txt

All the CSV files in the 'daily' directory are indeed compressed using the .gz format.

### (b) How many years are contained in daily, and how does the size of the data change?

From the year 1750 to the year 2025, there should be 276 years, but only 264 csv.gz files were found, missing daa for 12 years(1751-1762).
Generally, the size of the data increased yearly except for few years, from 10^{-6} MB to 10^{5} MB.
By the end of 2024, the size of the data reached to 10^{5} MB scale.

```
# find missing years
all_years = set(range(df["year"].min(), df["year"].max() + 1))
print("all year count:", len(all_years))

exisisting_years = set(df["year"])
print("GHCN daily dataset year count:", len(exisisting_years))

missing_year = sorted(all_years - exisisting_years)
print("Missing Years:", missing_year)
print("Missing Years Count:", len(missing_year))
```

```
all year count: 276
GHCN daily dataset year count: 264
Missing Years: [1751, 1752, 1753, 1754, 1755, 1756, 1757, 1758, 1759, 1760, 1761, 1762]
Missing Years Count: 12
```

### (c) What is the total size of the data, and how much of that is daily?

```
"""
Usage: hadoop fs -du [-s] [-h] [-v] [-x] URI [URI ...]
Displays sizes of files and directories contained in the given directory or the length of a file in case its just a
Options:
The -s option will result in an aggregate summary of file lengths being displayed, rather than the individual files.
The -h option will format file sizes in a "human-readable" fashion (e.g 64.0m instead of 67108864)
The -v option will display the names of columns as a header line.
The -x option will exclude snapshots from the result calculation. Without the -x option (default), the result is alw
"""

!hdfs dfs -du -s -h -v {DATA_ROOT}
!hdfs dfs -du -s -h -v {paths["daily"]}
```

```
SIZE    DISK_SPACE_CONSUMED_WITH_ALL_REPLICAS  FULL_PATH_NAME
13.1 G  13.1 G                                 wasbs://campus-data@madsstorage002.blob.core.windows.net/ghcnd
SIZE    DISK_SPACE_CONSUMED_WITH_ALL_REPLICAS  FULL_PATH_NAME
13.0 G  13.0 G                                 wasbs://campus-data@madsstorage002.blob.core.windows.net/ghcnd/daily
```

## Q2 DataType and Schema

You will now load each dataset to ensure the descriptions are accurate and that you can apply the schema either as the data is loaded or by casting columns as they are extracted by manually  rocessing the text records. Extend the code example in the notebook provided by following the steps below.

### (a) Define a schema for daily based on the description above or in the GHCN Daily README, using the types defined in pyspark.sql. What do you think is the best way to load the DATE and OBSERVATION TIME columns?

```
# Define daily_schema
# https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.types.StructField.html
daily_schema = StructType(
    [
        StructField("ID", StringType()),  # Character Station code
        StructField(
            "DATE", StringType()
        ),  # Date Observation date formatted as YYYYMMDD
        StructField("ELEMENT", StringType()),  # Character Element type indicator
        StructField("VALUE", DoubleType()),  # Real Data value for ELEMENT
```

```
        StructField("MEASUREMENT", StringType()),  # Character Measurement Flag
        StructField("QUALITY", StringType()),  # Character Quality Flag
        StructField("SOURCE", StringType()),  # Character Source Flag
        StructField("TIME", StringType()),  # Time Observation time formatted as HHMM
    ]
)

# load daily
daily = spark.read.csv(
    path=paths["daily"],
    schema=daily_schema,
)

print(type(daily))
daily.printSchema()
daily.show(20)
```

```
<class 'pyspark.sql.dataframe.DataFrame'>
root
 |-- ID: string (nullable = true)
 |-- DATE: string (nullable = true)
 |-- ELEMENT: string (nullable = true)
 |-- VALUE: double (nullable = true)
 |-- MEASUREMENT: string (nullable = true)
 |-- QUALITY: string (nullable = true)
 |-- SOURCE: string (nullable = true)
 |-- TIME: string (nullable = true)

+-----------+--------+-------+-----+-----------+-------+------+----+
|         ID|    DATE|ELEMENT|VALUE|MEASUREMENT|QUALITY|SOURCE|TIME|
+-----------+--------+-------+-----+-----------+-------+------+----+
|ASN00030019|20100101|   PRCP| 24.0|       NULL|   NULL|     a|NULL|
|ASN00030021|20100101|   PRCP|200.0|       NULL|   NULL|     a|NULL|
|ASN00030022|20100101|   TMAX|294.0|       NULL|   NULL|     a|NULL|
|ASN00030022|20100101|   TMIN|215.0|       NULL|   NULL|     a|NULL|
|ASN00030022|20100101|   PRCP|408.0|       NULL|   NULL|     a|NULL|
|ASN00029121|20100101|   PRCP|820.0|       NULL|   NULL|     a|NULL|
|ASN00029126|20100101|   TMAX|371.0|       NULL|   NULL|     S|NULL|
|ASN00029126|20100101|   TMIN|225.0|       NULL|   NULL|     S|NULL|
|ASN00029126|20100101|   PRCP|  0.0|       NULL|   NULL|     a|NULL|
|ASN00029126|20100101|   TAVG|298.0|          H|   NULL|     S|NULL|
|ASN00029127|20100101|   TMAX|371.0|       NULL|   NULL|     a|NULL|
|ASN00029127|20100101|   TMIN|225.0|       NULL|   NULL|     a|NULL|
|ASN00029127|20100101|   PRCP|  8.0|       NULL|   NULL|     a|NULL|
|ASN00029129|20100101|   PRCP|174.0|       NULL|   NULL|     a|NULL|
|ASN00029130|20100101|   PRCP| 86.0|       NULL|   NULL|     a|NULL|
|ASN00029131|20100101|   PRCP| 56.0|       NULL|   NULL|     a|NULL|
|ASN00029132|20100101|   PRCP|800.0|       NULL|   NULL|     a|NULL|
|ASN00029136|20100101|   PRCP| 22.0|       NULL|   NULL|     a|NULL|
|ASN00029137|20100101|   PRCP|  0.0|       NULL|   NULL|     a|NULL|
|ASN00029139|20100101|   TMAX|298.0|       NULL|   NULL|     a|NULL|
+-----------+--------+-------+-----+-----------+-------+------+----+
only showing top 20 rows
```

### (b) Modify the spark.read.csv command to load a subset of the most recent year of daily into Spark so that it uses the schema that you defined in step (a). Did anything go wrong when you tried to use the schema? What data types did you end up using and why?

```
latest_year = 2025
print(f"load year {latest_year} of daily and show its data schema")

daily_2025 = spark.read.csv(
    path=f"{DATA_ROOT}/daily/{latest_year}.csv.gz", schema=daily_schema
)

# parse DATE col
daily_2025 = daily_2025.withColumn("DATE", F.to_date(F.col("DATE"), "yyyyMMdd"))

# parse TIME
# 1)  DATE: YYYYMMDD -> DateType
daily_2025 = daily_2025.withColumn("DATE", F.to_date(F.col("DATE"), "yyyyMMdd"))

# 2) process TIME col: Null/space/4bit, clean and lpad to 4bit
daily_2025 = daily_2025.withColumn(
    "TIME_CLEAN",
    F.when(F.col("TIME").isNull() | (F.trim(F.col("TIME")) == ""), None)
     .otherwise(F.lpad(F.col("TIME").cast("string"), 4, "0"))
)
```

```

# 3) combine “virture date 19700101 + HHmm” and convert to  TimestampType
daily_2025 = daily_2025.withColumn(
    "OBS_TS",
    F.when(F.col("TIME_CLEAN").isNull(), None)
     .otherwise(F.to_timestamp(
         F.concat(F.lit("1970-01-01 "), F.col("TIME_CLEAN")),  # -> "1970-01-01 HHmm"
         "yyyy-MM-dd HHmm"
     ))
)

# 4) "HH:mm" string col for show
daily_2025 = daily_2025.withColumn(
    "OBS_HHMM",
    F.when(F.col("OBS_TS").isNull(), None)
     .otherwise(F.date_format(F.col("OBS_TS"), "HH:mm"))
)

# show the sample of data and print the schema
print(f"type of daily_latest variable:", type(daily_2025))
daily_2025.printSchema()
show_as_html(daily_2025)
```

```
load year 2025 of daily and show its data schema
type of daily_latest variable: <class 'pyspark.sql.dataframe.DataFrame'>
root
 |-- ID: string (nullable = true)
 |-- DATE: date (nullable = true)
 |-- ELEMENT: string (nullable = true)
 |-- VALUE: double (nullable = true)
 |-- MEASUREMENT: string (nullable = true)
 |-- QUALITY: string (nullable = true)
 |-- SOURCE: string (nullable = true)
 |-- TIME: string (nullable = true)
 |-- TIME_CLEAN: string (nullable = true)
 |-- OBS_TS: timestamp (nullable = true)
 |-- OBS_HHMM: string (nullable = true)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

| ID | DATE | ELEMENT | VALUE | MEASUREMENT | QUALITY | SOURCE | TIME | TIME_CLEAN | OBS_TS | OBS_HH |
|---|---|---|---|---|---|---|---|---|---|---|
| ASN00030019 | 2025-01-01 | PRCP | 0.0 | None | None | a | None | None | NaT | None |
| ASN00030021 | 2025-01-01 | PRCP | 0.0 | None | None | a | None | None | NaT | None |
| ASN00030022 | 2025-01-01 | TMAX | 414.0 | None | None | a | None | None | NaT | None |
| ASN00030022 | 2025-01-01 | TMIN | 247.0 | None | None | a | None | None | NaT | None |
| ASN00030022 | 2025-01-01 | PRCP | 0.0 | None | None | a | None | None | NaT | None |
| ASN00030025 | 2025-01-01 | PRCP | 0.0 | None | None | a | None | None | NaT | None |
| ASN00029118 | 2025-01-01 | PRCP | 0.0 | None | None | a | None | None | NaT | None |
| ASN00029121 | 2025-01-01 | PRCP | 0.0 | None | None | a | None | None | NaT | None |
| ASN00029126 | 2025-01-01 | TMAX | 414.0 | None | None | S | None | None | NaT | None |
| ASN00029126 | 2025-01-01 | TMIN | 198.0 | None | None | S | None | None | NaT | None |
| ASN00029126 | 2025-01-01 | PRCP | 0.0 | None | None | a | None | None | NaT | None |
| ASN00029126 | 2025-01-01 | TAVG | 321.0 | H | None | S | None | None | NaT | None |
| ASN00029127 | 2025-01-01 | TMAX | 414.0 | None | None | a | None | None | NaT | None |
| ASN00029127 | 2025-01-01 | TMIN | 198.0 | None | None | a | None | None | NaT | None |
| ASN00029127 | 2025-01-01 | PRCP | 0.0 | None | None | a | None | None | NaT | None |
| ASN00029129 | 2025-01-01 | PRCP | 0.0 | None | None | a | None | None | NaT | None |
| ASN00029131 | 2025-01-01 | PRCP | 0.0 | None | None | a | None | None | NaT | None |
| ASN00029132 | 2025-01-01 | PRCP | 0.0 | None | None | a | None | None | NaT | None |
| ASN00029136 | 2025-01-01 | PRCP | 0.0 | None | None | a | None | None | NaT | None |
| ASN00029139 | 2025-01-01 | TMAX | 347.0 | None | None | a | None | None | NaT | None |

**(c) Load each of stations, states, countries, and inventory datasets into Spark and find a way to extract the columns and data types in the descriptions above. You will need to parse the fixed width text formatting by hand, as there is no method to load this format implemented in the standard spark.read library. You should use pyspark.sql.functions.substring to extract the columns based on their character range.**

```
# Q2c: parse fixed-width text (stations/countries/states/inventory)
"""
IV. FORMAT OF "ghcnd-stations.txt"
------------------------------
Variable   Columns   Type
------------------------------
ID            1-11   Character
LATITUDE     13-20   Real
LONGITUDE    22-30   Real
ELEVATION    32-37   Real
STATE        39-40   Character
NAME         42-71   Character
GSN FLAG     73-75   Character
HCN/CRN FLAG 77-79   Character
WMO ID       81-85   Character
------------------------------
"""

stations_raw = spark.read.text(paths["stations"])
stations_df = (
    stations_raw.withColumn("ID", F.substring("value", 1, 11))
    .withColumn("LATITUDE", F.substring("value", 13, 8).cast("double"))
    .withColumn("LONGITUDE", F.substring("value", 22, 9).cast("double"))
    .withColumn("ELEVATION", F.substring("value", 32, 6).cast("double"))
    .withColumn("STATE", F.substring("value", 39, 2))
    .withColumn("NAME", F.substring("value", 42, 30))
    .withColumn("GSN_FLAG", F.substring("value", 73, 3))
    .withColumn("HCN_CRN", F.substring("value", 77, 3))
    .withColumn("WMO_ID", F.substring("value", 81, 5))
    .drop("value")
)

show_as_html(stations_df)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

| ID | LATITUDE | LONGITUDE | ELEVATION | STATE | NAME | GSN_FLAG | HCN_CRN | WMO_ID |
|---|---|---|---|---|---|---|---|---|
| ACW00011604 | 17.1167 | -61.7833 | 10.1 | | ST JOHNS COOLIDGE FLD | | | |
| ACW00011647 | 17.1333 | -61.7833 | 19.2 | | ST JOHNS | | | |
| AE000041196 | 25.3330 | 55.5170 | 34.0 | | SHARJAH INTER. AIRP | GSN | | 41196 |
| AEM00041194 | 25.2550 | 55.3640 | 10.4 | | DUBAI INTL | | | 41194 |
| AEM00041217 | 24.4330 | 54.6510 | 26.8 | | ABU DHABI INTL | | | 41217 |
| AEM00041218 | 24.2620 | 55.6090 | 264.9 | | AL AIN INTL | | | 41218 |
| AF000040930 | 35.3170 | 69.0170 | 3366.0 | | NORTH-SALANG | GSN | | 40930 |
| AFM00040938 | 34.2100 | 62.2280 | 977.2 | | HERAT | | | 40938 |
| AFM00040948 | 34.5660 | 69.2120 | 1791.3 | | KABUL INTL | | | 40948 |
| AFM00040990 | 31.5000 | 65.8500 | 1010.0 | | KANDAHAR AIRPORT | | | 40990 |
| AG000060390 | 36.7167 | 3.2500 | 24.0 | | ALGER-DAR EL BEIDA | GSN | | 60390 |
| AG000060590 | 30.5667 | 2.8667 | 397.0 | | EL-GOLEA | GSN | | 60590 |
| AG000060611 | 28.0500 | 9.6331 | 561.0 | | IN-AMENAS | GSN | | 60611 |
| AG000060680 | 22.8000 | 5.4331 | 1362.0 | | TAMANRASSET | GSN | | 60680 |
| AGE00135039 | 35.7297 | 0.6500 | 50.0 | | ORAN-HOPITAL MILITAIRE | | | |
| AGE00147704 | 36.9700 | 7.7900 | 161.0 | | ANNABA-CAP DE GARDE | | | |
| AGE00147705 | 36.7800 | 3.0700 | 59.0 | | ALGIERS-VILLE/UNIVERSITE | | | |
| AGE00147706 | 36.8000 | 3.0300 | 344.0 | | ALGIERS-BOUZAREAH | | | |
| AGE00147707 | 36.8000 | 3.0400 | 38.0 | | ALGIERS-CAP CAXINE | | | |
| AGE00147708 | 36.7200 | 4.0500 | 222.0 | | TIZI OUZOU | | | 60395 |

```
"""
V. FORMAT OF "ghcnd-countries.txt"

------------------------------
Variable   Columns   Type
------------------------------
CODE          1-2    Character
NAME          4-64   Character
------------------------------

These variables have the following definitions:

CODE       is the FIPS country code of the country where the station is
           located (from FIPS Publication 10-4 at
           www.cia.gov/cia/publications/factbook/appendix/appendix-d.html).

NAME       is the name of the country.
"""
countries_df = (
    spark.read.text(paths["countries"])
    .withColumn("CODE", F.substring("value", 1, 2))
    .withColumn("COUNTRY_NAME", F.substring("value", 4, 61))
    .drop("value")
)

show_as_html(countries_df)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

| CODE | COUNTRY_NAME |
|---|---|
| AC | Antigua and Barbuda |
| AE | United Arab Emirates |
| AF | Afghanistan |
| AG | Algeria |
| AJ | Azerbaijan |
| AL | Albania |
| AM | Armenia |
| AO | Angola |
| AQ | American Samoa [United States] |
| AR | Argentina |
| AS | Australia |
| AU | Austria |
| AY | Antarctica |
| BA | Bahrain |
| BB | Barbados |
| BC | Botswana |
| BD | Bermuda [United Kingdom] |
| BE | Belgium |
| BF | Bahamas, The |
| BG | Bangladesh |

```
"""
VI. FORMAT OF "ghcnd-states.txt"

------------------------------
Variable   Columns   Type
------------------------------
CODE          1-2    Character
NAME         4-50    Character
------------------------------

These variables have the following definitions:

CODE       is the POSTAL code of the U.S. state/territory or Canadian
           province where the station is located

NAME       is the name of the state, territory or province.
"""


states_df = (
    spark.read.text(paths["states"])
    .withColumn("CODE", F.substring("value", 1, 2))
    .withColumn("STATE_NAME", F.substring("value", 4, 47))
    .drop("value")
)
show_as_html(states_df)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

| CODE | STATE_NAME |
|---|---|
| AB | ALBERTA |
| AK | ALASKA |
| AL | ALABAMA |
| AR | ARKANSAS |
| AS | AMERICAN SAMOA |
| AZ | ARIZONA |
| BC | BRITISH COLUMBIA |
| CA | CALIFORNIA |
| CO | COLORADO |
| CT | CONNECTICUT |
| DC | DISTRICT OF COLUMBIA |
| DE | DELAWARE |
| FL | FLORIDA |
| FM | MICRONESIA |
| GA | GEORGIA |
| GU | GUAM |
| HI | HAWAII |
| IA | IOWA |
| ID | IDAHO |
| IL | ILLINOIS |

```
"""
VII. FORMAT OF "ghcnd-inventory.txt"

------------------------------
Variable   Columns   Type
------------------------------
ID            1-11   Character
LATITUDE     13-20   Real
LONGITUDE    22-30   Real
ELEMENT      32-35   Character
FIRSTYEAR    37-40   Integer
LASTYEAR     42-45   Integer
------------------------------

These variables have the following definitions:

ID          is the station identification code.  Please see "ghcnd-stations.txt"
            for a complete list of stations and their metadata.

LATITUDE    is the latitude of the station (in decimal degrees).

LONGITUDE   is the longitude of the station (in decimal degrees).

ELEMENT     is the element type.  See section III for a definition of elements.

FIRSTYEAR   is the first year of unflagged data for the given element.

LASTYEAR    is the last year of unflagged data for the given element.
"""

inventory_df = (
    spark.read.text(paths["inventory"])
```

```
    .withColumn("ID", F.substring("value", 1, 11))
    .withColumn("LATITUDE", F.substring("value", 13, 8).cast("double"))
    .withColumn("LONGITUDE", F.substring("value", 22, 9).cast("double"))
    .withColumn("ELEMENT", F.substring("value", 32, 4))
    .withColumn("FIRSTYEAR", F.substring("value", 37, 4).cast("int"))
    .withColumn("LASTYEAR", F.substring("value", 42, 4).cast("int"))
    .drop("value")
)
show_as_html(inventory_df)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

| ID | LATITUDE | LONGITUDE | ELEMENT | FIRSTYEAR | LASTYEAR |
|---|---|---|---|---|---|
| ACW00011604 | 17.1167 | -61.7833 | TMAX | 1949 | 1949 |
| ACW00011604 | 17.1167 | -61.7833 | TMIN | 1949 | 1949 |
| ACW00011604 | 17.1167 | -61.7833 | PRCP | 1949 | 1949 |
| ACW00011604 | 17.1167 | -61.7833 | SNOW | 1949 | 1949 |
| ACW00011604 | 17.1167 | -61.7833 | SNWD | 1949 | 1949 |
| ACW00011604 | 17.1167 | -61.7833 | PGTM | 1949 | 1949 |
| ACW00011604 | 17.1167 | -61.7833 | WDFG | 1949 | 1949 |
| ACW00011604 | 17.1167 | -61.7833 | WSFG | 1949 | 1949 |
| ACW00011604 | 17.1167 | -61.7833 | WT03 | 1949 | 1949 |
| ACW00011604 | 17.1167 | -61.7833 | WT08 | 1949 | 1949 |
| ACW00011604 | 17.1167 | -61.7833 | WT16 | 1949 | 1949 |
| ACW00011647 | 17.1333 | -61.7833 | TMAX | 1961 | 1961 |
| ACW00011647 | 17.1333 | -61.7833 | TMIN | 1961 | 1961 |
| ACW00011647 | 17.1333 | -61.7833 | PRCP | 1957 | 1970 |
| ACW00011647 | 17.1333 | -61.7833 | SNOW | 1957 | 1970 |
| ACW00011647 | 17.1333 | -61.7833 | SNWD | 1957 | 1970 |
| ACW00011647 | 17.1333 | -61.7833 | WT03 | 1961 | 1961 |
| ACW00011647 | 17.1333 | -61.7833 | WT16 | 1961 | 1966 |
| AE000041196 | 25.3330 | 55.5170 | TMAX | 1944 | 2025 |
| AE000041196 | 25.3330 | 55.5170 | TMIN | 1944 | 2025 |

## (d) How many rows are there in each of the metadata tables?

```
# Q2d–e: rows count
meta_counts = {
    "stations": stations_df.count(),
    "countries": countries_df.count(),
    "states": states_df.count(),
    "inventory": inventory_df.count(),
    "daily": daily.count(),
}

meta_counts
```

```
{'stations': 129657,
 'countries': 219,
 'states': 74,
 'inventory': 766784,
 'daily': 3155140380}
```

```
# save meta counts to txt file
with open("./supplementary/meta_counts.txt",mode= "w") as f:
    for k,v in meta_counts.items():
        f.write(f"{k}:{v}\n")
```

### (e) How many rows are there in daily?

Note that this will take a while if you are only using 2 executors and 1 core per executor, and that the amount of driver and executor memory should not matter unless you actually try to cache or collect all of daily. You should not try to cache or collect all of daily.

```
meta_counts.get("daily")
```

```
3155140380
```

## Q3 Metadata Tables Combination

Next you will combine relevant information from the metadata tables by joining on station, state, and country to give an enriched stations table that we can use for filtering based on attributes at a station level.

### (a) Extract the two character country code from each station code in stations and store the output as a new column using the withColumn method.

```
# Q3a: extract station ID
stations_enriched = stations_df.withColumn("COUNTRY_CODE", F.substring("ID", 1, 2))
show_as_html(stations_enriched,5)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

| ID | LATITUDE | LONGITUDE | ELEVATION | STATE | NAME | GSN_FLAG | HCN_CRN | WMO_ID | COUNTRY_CODE |
|---|---|---|---|---|---|---|---|---|---|
| ACW00011604 | 17.1167 | -61.7833 | 10.1 | | ST JOHNS COOLIDGE FLD | | | | AC |
| ACW00011647 | 17.1333 | -61.7833 | 19.2 | | ST JOHNS | | | | AC |
| AE000041196 | 25.3330 | 55.5170 | 34.0 | | SHARJAH INTER. AIRP | GSN | | 41196 | AE |
| AEM00041194 | 25.2550 | 55.3640 | 10.4 | | DUBAI INTL | | | 41194 | AE |
| AEM00041217 | 24.4330 | 54.6510 | 26.8 | | ABU DHABI INTL | | | 41217 | AE |

### (b) LEFT JOIN stations with countries using your output from step (a).

```
# Q3b: LEFT JOIN countries
stations_enriched = stations_enriched.join(
    countries_df.withColumnRenamed(
        "CODE", "COUNTRY_CODE"
    ),  # rename countries_df col as the same as station_enriched col name before on = "colname" called
    on="COUNTRY_CODE",
    how="left",
)
show_as_html(stations_enriched,5)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

| COUNTRY_CODE | ID | LATITUDE | LONGITUDE | ELEVATION | STATE | NAME | GSN_FLAG | HCN_CRN | WMO_ID |
|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| AC | ACW00011604 | 17.1167 | -61.7833 | 10.1 | | ST JOHNS COOLIDGE FLD | | | |
| AC | ACW00011647 | 17.1333 | -61.7833 | 19.2 | | ST JOHNS | | | |
| AE | AE000041196 | 25.3330 | 55.5170 | 34.0 | | SHARJAH INTER. AIRP | GSN | | 41196 |
| AE | AEM00041194 | 25.2550 | 55.3640 | 10.4 | | DUBAI INTL | | | 41194 |
| AE | AEM00041217 | 24.4330 | 54.6510 | 26.8 | | ABU DHABI INTL | | | 41217 |

## (c) LEFT JOIN stations and states, allowing for the fact that state codes are only provided for stations in the US.

```
show_as_html(states_df,2)

# check stations_enriched non-empty col "STATE" status
stations_enriched.filter(F.trim(F.col("STATE")).isNotNull() & (F.trim(F.col("STATE"))!="")).show()

```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

| CODE | STATE_NAME |
|---|---|
| AB | ALBERTA |
| AK | ALASKA |

```
+------------+-----------+--------+---------+---------+-----+--------------------+--------+-------+------+--------------------+
|COUNTRY_CODE|         ID|LATITUDE|LONGITUDE|ELEVATION|STATE|                NAME|GSN_FLAG|HCN_CRN|WMO_ID|        COUNTRY_NAME|
+------------+-----------+--------+---------+---------+-----+--------------------+--------+-------+------+--------------------+
|          AQ|AQC00914000|-14.3167|-170.7667|    408.4|   AS|AASUFOU          ...|        |       |      |American Samoa [U...|
|          AQ|AQC00914005|-14.2667|  -170.65|    182.9|   AS|AFONO            ...|        |       |      |American Samoa [U...|
|          AQ|AQC00914021|-14.2667|-170.5833|      6.1|   AS|AMOULI TUTUILA   ...|        |       |      |American Samoa [U...|
|          AQ|AQC00914060|-14.2667|-170.6833|     80.8|   AS|ATUU             ...|        |       |      |American Samoa [U...|
|          AQ|AQC00914135|  -14.3|  -170.7|    249.9|   AS|FAGA ALU RSVR    ...|        |       |      |American Samoa [U...|
|          AQ|AQC00914138|-14.2833|-170.6833|     24.1|   AS|FAGA ALU STREAM  ...|        |       |      |American Samoa [U...|
|          AQ|AQC00914141|-14.2667|-170.6167|      4.6|   AS|FAGAITUA         ...|        |       |      |American Samoa [U...|
|          AQ|AQC00914145|-14.2833|-170.7167|     14.9|   AS|FAGASA TUTUILA   ...|        |       |      |American Samoa [U...|
|          AQ|AQC00914149|-14.2833|-170.6833|     57.0|   AS|FAGA TOGO        ...|        |       |      |American Samoa [U...|
|          AQ|AQC00914188|-14.2167|-168.5333|      6.1|   AS|FALEASAO TAU     ...|        |       |      |American Samoa [U...|
|          AQ|AQC00914248|-14.2333|-169.5167|      4.9|   AS|FALEASAO VILLAGE ...|        |       |      |American Samoa [U...|
|          AQ|AQC00914397|  -14.35|-170.7833|      6.1|   AS|LEONE            ...|        |       |      |American Samoa [U...|
|          AQ|AQC00914424|-14.2333|-169.5167|      6.1|   AS|LUMA VILLAGE     ...|        |       |      |American Samoa [U...|
|          AQ|AQC00914594|-14.3333|-170.7667|     42.4|   AS|MALAELOA         ...|        |       |      |American Samoa [U...|
|          AQ|AQC00914650|-14.1667|-169.7167|      3.0|   AS|OFU              ...|        |       |      |American Samoa [U...|
|          AQ|AQC00914822|  -11.05|-171.0833|      3.0|   AS|SWAIN ISLAND     ...|        |       |      |American Samoa [U...|
|          AQ|AQC00914869|-14.3333|-170.7167|      3.0|   AS|TAFUNA AP TUTUILA...|        |       |      |American Samoa [U...|
|          AQ|AQC00914873|  -14.35|-170.7667|     14.9|   AS|TAPUTIMU TUTUILA ...|        |       |      |American Samoa [U...|
|          AQ|AQC00914902|-14.2728|-170.6922|     80.8|   AS|VAIPITO          ...|        |       |      |American Samoa [U...|
|          AQ|AQC00914912|  -14.25|-170.6667|      3.0|   AS|VATIA            ...|        |       |      |American Samoa [U...|
+------------+-----------+--------+---------+---------+-----+--------------------+--------+-------+------+--------------------+
only showing top 20 rows
```

```
# as we already check above cell, we should rename states_df col "CODE" to col "STATE" as to match
# station_enriched col "STATE"
stations_enriched = stations_enriched.join(
    states_df.withColumnRenamed("CODE", "STATE"), on="STATE", how="left"
)

show_as_html(stations_enriched)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

| STATE | COUNTRY_CODE | ID | LATITUDE | LONGITUDE | ELEVATION | NAME | GSN_FLAG | HCN_CRN | WMO_ID |
|---|---|---|---|---|---|---|---|---|---|
| | AC | ACW00011604 | 17.1167 | -61.7833 | 10.1 | ST JOHNS COOLIDGE FLD | | | |
| | AC | ACW00011647 | 17.1333 | -61.7833 | 19.2 | ST JOHNS | | | |
| | AE | AE000041196 | 25.3330 | 55.5170 | 34.0 | SHARJAH INTER. AIRP | GSN | | 41196 |
| | AE | AEM00041194 | 25.2550 | 55.3640 | 10.4 | DUBAI INTL | | | 41194 |
| | AE | AEM00041217 | 24.4330 | 54.6510 | 26.8 | ABU DHABI INTL | | | 41217 |
| | AE | AEM00041218 | 24.2620 | 55.6090 | 264.9 | AL AIN INTL | | | 41218 |
| | AF | AF000040930 | 35.3170 | 69.0170 | 3366.0 | NORTH-SALANG | GSN | | 40930 |
| | AF | AFM00040938 | 34.2100 | 62.2280 | 977.2 | HERAT | | | 40938 |
| | AF | AFM00040948 | 34.5660 | 69.2120 | 1791.3 | KABUL INTL | | | 40948 |
| | AF | AFM00040990 | 31.5000 | 65.8500 | 1010.0 | KANDAHAR AIRPORT | | | 40990 |
| | AG | AG000060390 | 36.7167 | 3.2500 | 24.0 | ALGER-DAR EL BEIDA | GSN | | 60390 |
| | AG | AG000060590 | 30.5667 | 2.8667 | 397.0 | EL-GOLEA | GSN | | 60590 |
| | AG | AG000060611 | 28.0500 | 9.6331 | 561.0 | IN-AMENAS | GSN | | 60611 |
| | AG | AG000060680 | 22.8000 | 5.4331 | 1362.0 | TAMANRASSET | GSN | | 60680 |
| | AG | AGE00135039 | 35.7297 | 0.6500 | 50.0 | ORAN-HOPITAL MILITAIRE | | | |
| | AG | AGE00147704 | 36.9700 | 7.7900 | 161.0 | ANNABA-CAP DE GARDE | | | |
| | AG | AGE00147705 | 36.7800 | 3.0700 | 59.0 | ALGIERS-VILLE/UNIVERSITE | | | |
| | AG | AGE00147706 | 36.8000 | 3.0300 | 344.0 | ALGIERS-BOUZAREAH | | | |
| | AG | AGE00147707 | 36.8000 | 3.0400 | 38.0 | ALGIERS-CAP CAXINE | | | |
| | AG | AGE00147708 | 36.7200 | 4.0500 | 222.0 | TIZI OUZOU | | | 60395 |

## (d) Based on inventory, what was the first and last year that each station was active and collected any element at all?

How many different elements has each station collected overall?
Further, count separately the number of core elements and the number of ”other” elements that each station has collected overall.
How many stations collect all five core elements?
How many collect only precipitation and no other elements?

Note that we could also determine the set of elements that each station has collected and store this output as a new column using pyspark.sql.functions.collect set but it will be more efficient to first filter inventory by element type using the element column and then to join against that output as necessary.

```
show_as_html(inventory_df)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

| ID | LATITUDE | LONGITUDE | ELEMENT | FIRSTYEAR | LASTYEAR |
|---|---|---|---|---|---|
| ACW00011604 | 17.1167 | -61.7833 | TMAX | 1949 | 1949 |
| ACW00011604 | 17.1167 | -61.7833 | TMIN | 1949 | 1949 |
| ACW00011604 | 17.1167 | -61.7833 | PRCP | 1949 | 1949 |
| ACW00011604 | 17.1167 | -61.7833 | SNOW | 1949 | 1949 |
| ACW00011604 | 17.1167 | -61.7833 | SNWD | 1949 | 1949 |
| ACW00011604 | 17.1167 | -61.7833 | PGTM | 1949 | 1949 |
| ACW00011604 | 17.1167 | -61.7833 | WDFG | 1949 | 1949 |
| ACW00011604 | 17.1167 | -61.7833 | WSFG | 1949 | 1949 |
| ACW00011604 | 17.1167 | -61.7833 | WT03 | 1949 | 1949 |

| | | | | | |
|---|---|---|---|---|---|
| ACW00011604 | 17.1167 | -61.7833 | WT08 | 1949 | 1949 |
| ACW00011604 | 17.1167 | -61.7833 | WT16 | 1949 | 1949 |
| ACW00011647 | 17.1333 | -61.7833 | TMAX | 1961 | 1961 |
| ACW00011647 | 17.1333 | -61.7833 | TMIN | 1961 | 1961 |
| ACW00011647 | 17.1333 | -61.7833 | PRCP | 1957 | 1970 |
| ACW00011647 | 17.1333 | -61.7833 | SNOW | 1957 | 1970 |
| ACW00011647 | 17.1333 | -61.7833 | SNWD | 1957 | 1970 |
| ACW00011647 | 17.1333 | -61.7833 | WT03 | 1961 | 1961 |
| ACW00011647 | 17.1333 | -61.7833 | WT16 | 1961 | 1966 |
| AE000041196 | 25.3330 | 55.5170 | TMAX | 1944 | 2025 |
| AE000041196 | 25.3330 | 55.5170 | TMIN | 1944 | 2025 |

```python
# 1) Basic station-level statistics (without using collect_set)
base_stats = (
    inventory_df
    .groupBy("ID")
    .agg(
        F.min("FIRSTYEAR").alias("FIRSTYEAR_ANY"),
        F.max("LASTYEAR").alias("LASTYEAR_ANY"),
        F.countDistinct("ELEMENT").alias("N_ELEMENTS")
    )
)


# 2) Filter by element type first (as suggested in the assignment)
#    Create one-hot indicator columns for the core elements
core = ["TMAX", "TMIN", "PRCP", "SNOW", "SNWD"]

# Keep only core elements and deduplicate by (station, element)
core_pairs = (
    inventory_df
    .filter(F.col("ELEMENT").isin(core))
    .select("ID", "ELEMENT")
    .distinct()
)

# Pivot to wide format (each core element becomes a 0/1 indicator)
# Note: use count(*) > 0 → converted to 1/0
core_flags = (
    core_pairs
    .groupBy("ID")
    .pivot("ELEMENT", core)
    .agg(F.count(F.lit(1)))
    .na.fill(0)
)

# Optionally: convert counts to explicit 0/1 (can be omitted if already 0/1)
for el in core:
    core_flags = core_flags.withColumn(el, (F.col(el) > 0).cast("int"))

# ------------------------------
# 3) Merge: base stats + core element indicators
# ------------------------------
inv_by_station = (
    base_stats
    .join(core_flags, on="ID", how="left")
    .na.fill(0, subset=core)  # fill missing core elements with 0
    .withColumn("N_CORE_ELEMENTS", sum(F.col(el) for el in core))
)

show_as_html(inv_by_station)
```

```css
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

| ID | FIRSTYEAR_ANY | LASTYEAR_ANY | N_ELEMENTS | TMAX | TMIN | PRCP | SNOW | SNWD | N_CORE_ELEMENTS |
|---|---|---|---|---|---|---|---|---|---|
| AE000041196 | 1944 | 2025 | 4 | 1 | 1 | 1 | 0 | 0 | 3 |
| AEM00041218 | 1994 | 2025 | 4 | 1 | 1 | 1 | 0 | 0 | 3 |
| AGE00147715 | 1879 | 1938 | 3 | 1 | 1 | 1 | 0 | 0 | 3 |
| AGE00147717 | 1880 | 1938 | 3 | 1 | 1 | 1 | 0 | 0 | 3 |
| AGE00147719 | 1888 | 2025 | 4 | 1 | 1 | 1 | 0 | 0 | 3 |
| AGM00060425 | 1943 | 2025 | 5 | 1 | 1 | 1 | 0 | 1 | 4 |
| AGM00060430 | 1957 | 2025 | 5 | 1 | 1 | 1 | 0 | 1 | 4 |
| AGM00060490 | 1957 | 2025 | 5 | 1 | 1 | 1 | 0 | 1 | 4 |
| AGM00060514 | 1995 | 2025 | 5 | 1 | 1 | 1 | 0 | 1 | 4 |
| AGM00060515 | 1984 | 2025 | 4 | 1 | 1 | 1 | 0 | 0 | 3 |
| AGM00060522 | 1976 | 2025 | 5 | 1 | 1 | 1 | 0 | 1 | 4 |
| AGM00060536 | 1981 | 2025 | 5 | 1 | 1 | 1 | 0 | 1 | 4 |
| AGM00060555 | 1958 | 2025 | 4 | 1 | 1 | 1 | 0 | 0 | 3 |
| AGM00060580 | 1957 | 2025 | 4 | 1 | 1 | 1 | 0 | 0 | 3 |
| AJ000037742 | 1955 | 1987 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| AJ000037747 | 1955 | 2025 | 5 | 1 | 1 | 1 | 0 | 1 | 4 |
| AJ000037756 | 1955 | 2025 | 5 | 1 | 1 | 1 | 0 | 1 | 4 |
| AJ000037816 | 1963 | 1991 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| AJ000037899 | 1936 | 1991 | 5 | 1 | 1 | 1 | 0 | 1 | 4 |
| AJ000037914 | 1955 | 1991 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |

```
# # How many stations collect all five core elements?
n_all_five_core = inv_by_station.filter(F.col("N_CORE_ELEMENTS") == 5).count()

n_all_five_core
```

```
20504
```

```
# How many stations collect only preciptitation and no other elements?
# only PRCP and no other elements
n_prcp_only = inv_by_station.filter(
    (F.col("N_ELEMENTS") == 1) & (F.col("PRCP") == 1)
).count()

n_prcp_only
```

```
16267
```

```
stations_enriched = stations_enriched.join(
    inv_by_station.withColumnRenamed("ID", "ID"), on="ID", how="left"
)

show_as_html(stations_enriched)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

| ID | STATE | COUNTRY_CODE | LATITUDE | LONGITUDE | ELEVATION | NAME | GSN_FLAG | HCN_CRN | WMO_ID |
|---|---|---|---|---|---|---|---|---|---|
| AE000041196 | | AE | 25.3330 | 55.5170 | 34.0 | SHARJAH INTER. AIRP | GSN | | 41196 |
| AEM00041218 | | AE | 24.2620 | 55.6090 | 264.9 | AL AIN INTL | | | 41218 |
| RSM00030934 | | RS | 50.6000 | 107.5830 | 638.0 | BICURA | | | 30934 |
| US1VAAP0001 | VA | US | 37.4083 | -78.9638 | 219.5 | CONCORD 4.6 NNE | | | |
| US1VAAR0018 | VA | US | 38.8719 | -77.1190 | 78.0 | ARLINGTON 1.0 WSW | | | |
| US1VABC0003 | VA | US | 37.2187 | -82.1094 | 367.3 | VANSANT 1.4 SW | | | |
| US1VABD0010 | VA | US | 37.2762 | -79.8406 | 359.7 | VINTON 2.6 E | | | |
| AG000060590 | | AG | 30.5667 | 2.8667 | 397.0 | EL-GOLEA | GSN | | 60590 |
| AGE00147705 | | AG | 36.7800 | 3.0700 | 59.0 | ALGIERS-VILLE/UNIVERSITE | | | |
| AGE00147706 | | AG | 36.8000 | 3.0300 | 344.0 | ALGIERS-BOUZAREAH | | | |
| RSM00030856 | | RS | 51.4830 | 114.5330 | 877.0 | SEDLOVAJA | | | 30856 |
| US1VAAM0002 | VA | US | 37.3009 | -78.1609 | 118.0 | JETERSVILLE 3.6 W | | | |
| US1VAAR0005 | VA | US | 38.8554 | -77.0698 | 41.5 | ALEXANDRIA 2.4 NE | | | |
| US1VAAR0016 | VA | US | 38.8920 | -77.1617 | 109.7 | ARLINGTON 3.3 WNW | | | |
| AF000040930 | | AF | 35.3170 | 69.0170 | 3366.0 | NORTH-SALANG | GSN | | 40930 |
| AGE00147704 | | AG | 36.9700 | 7.7900 | 161.0 | ANNABA-CAP DE GARDE | | | |
| AGE00147708 | | AG | 36.7200 | 4.0500 | 222.0 | TIZI OUZOU | | | 60395 |
| RSM00030862 | | RS | 51.8670 | 116.0330 | 597.0 | SHILKA | | | 30862 |
| RSM00030935 | | RS | 50.3700 | 108.7500 | 770.0 | KRASNYJ CHIKOJ | | | 30935 |
| US1VAAR0017 | VA | US | 38.7986 | -77.0431 | 10.4 | HUNTINGTON 1.6 ENE | | | |

20 rows × 21 columns

## (e) save combined table stations_enriched in parquet format

```
stations_enriched.count()
size_bytes = stations_enriched.rdd.map(lambda row: len(str(row))).sum()
print("Approx size in MB:", size_bytes / (1024 * 1024))
```

```
[Stage 153:=======================================>                 (6 + 3) / 9]

Approx size in MB: 43.822248458862305
```

```
stations_enriched_savepath = USER_ROOT + "stations_enriched_parquet"
stations_enriched_savepath
# Save the stations_enriched to savepah
stations_enriched.write.mode("overwrite").parquet(stations_enriched_savepath)
```

```
# check save result
!hdfs dfs -ls -h {stations_enriched_savepath}
```

```
Found 10 items
-rw-r--r--   1 yxi75 supergroup          0 2025-09-11 18:30 wasbs://campus-user@madsstorage002.blob.core.windows.net/yxi75/stations_enriched_
-rw-r--r--   1 yxi75 supergroup    606.8 K 2025-09-11 18:30 wasbs://campus-user@madsstorage002.blob.core.windows.net/yxi75/stations_enriched_
-rw-r--r--   1 yxi75 supergroup    582.7 K 2025-09-11 18:30 wasbs://campus-user@madsstorage002.blob.core.windows.net/yxi75/stations_enriched_
-rw-r--r--   1 yxi75 supergroup    447.8 K 2025-09-11 18:30 wasbs://campus-user@madsstorage002.blob.core.windows.net/yxi75/stations_enriched_
-rw-r--r--   1 yxi75 supergroup    579.3 K 2025-09-11 18:30 wasbs://campus-user@madsstorage002.blob.core.windows.net/yxi75/stations_enriched_
-rw-r--r--   1 yxi75 supergroup    459.8 K 2025-09-11 18:30 wasbs://campus-user@madsstorage002.blob.core.windows.net/yxi75/stations_enriched_
-rw-r--r--   1 yxi75 supergroup    457.0 K 2025-09-11 18:30 wasbs://campus-user@madsstorage002.blob.core.windows.net/yxi75/stations_enriched_
-rw-r--r--   1 yxi75 supergroup    582.2 K 2025-09-11 18:30 wasbs://campus-user@madsstorage002.blob.core.windows.net/yxi75/stations_enriched_
-rw-r--r--   1 yxi75 supergroup    589.7 K 2025-09-11 18:30 wasbs://campus-user@madsstorage002.blob.core.windows.net/yxi75/stations_enriched_
-rw-r--r--   1 yxi75 supergroup    452.8 K 2025-09-11 18:30 wasbs://campus-user@madsstorage002.blob.core.windows.net/yxi75/stations_enriched_
```

## Q4 Check for missing stations in daily

```python
# load stations_enriched from Azure storage and cache it for frequently usage
stations_enriched = spark.read.parquet(stations_enriched_savepath)
stations_enriched.cache()
show_as_html(stations_enriched)
```

```css
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

| ID | STATE | COUNTRY_CODE | LATITUDE | LONGITUDE | ELEVATION | NAME | GSN_FLAG | HCN_CRN | WM |
|---|---|---|---|---|---|---|---|---|---|
| AE000041196 | | AE | 25.3330 | 55.5170 | 34.0 | SHARJAH INTER. AIRP | GSN | | 4119 |
| AEM00041218 | | AE | 24.2620 | 55.6090 | 264.9 | AL AIN INTL | | | 4121 |
| AGE00147715 | | AG | 35.4200 | 8.1197 | 863.0 | TEBESSA | | | |
| AGE00147717 | | AG | 35.2000 | 0.6300 | 476.0 | SIDI-BEL-ABBES | | | |
| AGE00147719 | | AG | 33.7997 | 2.8900 | 767.0 | LAGHOUAT | | | 6054 |
| AGM00060425 | | AG | 36.2130 | 1.3320 | 141.1 | ECH CHELIFF | | | 6042 |
| AGM00060430 | | AG | 36.3000 | 2.2330 | 721.0 | MILIANA | | | 6043 |
| AGM00060490 | | AG | 35.6240 | -0.6210 | 89.9 | ES SENIA | | | 6049 |
| AGM00060514 | | AG | 35.1670 | 2.3170 | 801.0 | KSAR CHELLALA | | | 6051 |
| AGM00060515 | | AG | 35.3330 | 4.2060 | 459.0 | BOU SAADA | | | 6051 |
| AGM00060522 | | AG | 34.8200 | -1.7700 | 426.0 | MAGHNIA | | | 6052 |
| AGM00060536 | | AG | 34.8670 | 0.1500 | 752.0 | SAIDA | | | 6053 |
| AGM00060555 | | AG | 33.0680 | 6.0890 | 85.0 | SIDI MAHDI | | | 6055 |
| AGM00060580 | | AG | 31.9170 | 5.4130 | 150.0 | OUARGLA | | | 6058 |
| AJ000037742 | | AJ | 40.9000 | 47.3000 | 313.0 | ORDJONIKIDZE,ZERNOSOVHOZ | | | 3774 |
| AJ000037747 | | AJ | 40.6170 | 47.1500 | 15.0 | EVLAKH AIRPORT | | | 3774 |
| AJ000037756 | | AJ | 40.5330 | 48.9330 | 755.0 | MARAZA | | | 3775 |
| AJ000037816 | | AJ | 40.5000 | 46.1000 | 1655.0 | DASHKESAN | | | 3781 |
| AJ000037899 | | AJ | 39.7670 | 46.7500 | 1355.0 | SHUSHA | | | 3789 |
| AJ000037914 | | AJ | 39.9000 | 48.0000 | -1.0 | IMISLY | | | 3791 |

20 rows × 21 columns

```python
# show loaded daily_2025,  a subset of daily to check missing id algorithm
show_as_html(daily_2025)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

| ID | DATE | ELEMENT | VALUE | MEASUREMENT | QUALITY | SOURCE | TIME | TIME_CLEAN | OBS_TS | OBS_HHMM |
|---|---|---|---|---|---|---|---|---|---|---|
| ASN00030019 | 2025-01-01 | PRCP | 0.0 | None | None | a | None | None | NaT | None |
| ASN00030021 | 2025-01-01 | PRCP | 0.0 | None | None | a | None | None | NaT | None |
| ASN00030022 | 2025-01-01 | TMAX | 414.0 | None | None | a | None | None | NaT | None |
| ASN00030022 | 2025-01-01 | TMIN | 247.0 | None | None | a | None | None | NaT | None |
| ASN00030022 | 2025-01-01 | PRCP | 0.0 | None | None | a | None | None | NaT | None |
| ASN00030025 | 2025-01-01 | PRCP | 0.0 | None | None | a | None | None | NaT | None |
| ASN00029118 | 2025-01-01 | PRCP | 0.0 | None | None | a | None | None | NaT | None |
| ASN00029121 | 2025-01-01 | PRCP | 0.0 | None | None | a | None | None | NaT | None |
| ASN00029126 | 2025-01-01 | TMAX | 414.0 | None | None | S | None | None | NaT | None |
| ASN00029126 | 2025-01-01 | TMIN | 198.0 | None | None | S | None | None | NaT | None |
| ASN00029126 | 2025-01-01 | PRCP | 0.0 | None | None | a | None | None | NaT | None |
| ASN00029126 | 2025-01-01 | TAVG | 321.0 | H | None | S | None | None | NaT | None |
| ASN00029127 | 2025-01-01 | TMAX | 414.0 | None | None | a | None | None | NaT | None |
| ASN00029127 | 2025-01-01 | TMIN | 198.0 | None | None | a | None | None | NaT | None |
| ASN00029127 | 2025-01-01 | PRCP | 0.0 | None | None | a | None | None | NaT | None |
| ASN00029129 | 2025-01-01 | PRCP | 0.0 | None | None | a | None | None | NaT | None |
| ASN00029131 | 2025-01-01 | PRCP | 0.0 | None | None | a | None | None | NaT | None |
| ASN00029132 | 2025-01-01 | PRCP | 0.0 | None | None | a | None | None | NaT | None |
| ASN00029136 | 2025-01-01 | PRCP | 0.0 | None | None | a | None | None | NaT | None |
| ASN00029139 | 2025-01-01 | TMAX | 347.0 | None | None | a | None | None | NaT | None |

```
# extract year 2025 daily show up station IDs (scalable in yera range)
daily_station_ids_2025 = (
    daily_2025.select("ID").distinct().withColumnRenamed("ID", "ID_IN_DAILY")
)

# use left anti join to find missing stations in daily_2025
missing_in_daily_2025 = stations_enriched.join(
    daily_station_ids_2025,
    stations_enriched.ID == daily_station_ids_2025.ID_IN_DAILY,
    how="left_anti",
)

missing_stations_count_2025 = missing_in_daily_2025.count()
missing_stations_count_2025
```

```
89941
```

```
# extract all the year daily show up station IDs (expand to all yera range)
daily_station_ids_total = (
    daily.select("ID").distinct().withColumnRenamed("ID", "ID_IN_DAILY")
)

# use left anti join to find missing stations in daily all years
missing_in_daily_total = stations_enriched.join(
    daily_station_ids_total,
```

```
    stations_enriched.ID == daily_station_ids_total.ID_IN_DAILY,
    how="left_anti",
)

missing_stations_count_total = missing_in_daily_total.count()
missing_stations_count_total
```

```
38
```

```
# save to supplementary
with open("./supplementary/missing_in_daily.txt","w") as f:
    f.write(f"missing_stations_count_2025:{missing_stations_count_2025}\n")
    f.write(f"missing_stations_count_total:{missing_stations_count_total}\n")

```

```
stop_spark()
```

```
25/09/11 19:26:51 WARN ExecutorPodsWatchSnapshotSource: Kubernetes client has been closed.
```

**Spark**

The spark session is **stopped**, confirm that `yxi75 (notebook)` is under the completed applications section in the Spark UI.

- Spark UI