```
# 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 22:13:42 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.app.id	spark-1054e131d02d4bc6a6e42e1ed4b1e027
spark.kubernetes.executor.podNamePrefix	yxi75-notebook-344ce89938444bdb
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.submitTime	1757585623019
spark.submit.deployMode	client
spark.master	k8s://https://kubernetes.default.svc.cluster.local:443
spark.driver.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 -Dderby.system.home=/tmp/yxi75/spark/
spark.fs.azure	org.apache.hadoop.fs.azure.NativeAzureFileSystem
spark.app.startTime	1757585623186
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.kubernetes.memoryOverheadFactor	0.3
spark.driver.host	spark-master-svc
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"
station_count_by_country_path = USER_ROOT + "station_count_by_country_parquet"
station_count_us_terri_path = USER_ROOT + "station_count_us_terri_parquet"
country_meta_with_station_num_path = USER_ROOT + "country_meta_with_station_num"
states_meta_with_station_num_path = USER_ROOT + "states_meta_with_station_num"
```

Analysis

Q1 Station Study

(a)

```
# load stations_enriched from saved path
stations_enriched = spark.read.parquet(stations_enriched_savepath)
stations_enriched.cache()
# check variable
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
AE000041196		AE	25.3330	55.5170	34.0	SHARJAH INTER. AIRP	GSN	
AEM00041218		AE	24.2620	55.6090	264.9	AL AIN INTL		
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		
AGM00060425		AG	36.2130	1.3320	141.1	ECH CHELIFF		
AGM00060430		AG	36.3000	2.2330	721.0	MILIANA		
AGM00060490		AG	35.6240	-0.6210	89.9	ES SENIA		
AGM00060514		AG	35.1670	2.3170	801.0	KSAR CHELLALA		
AGM00060515		AG	35.3330	4.2060	459.0	BOU SAADA		
AGM00060522		AG	34.8200	-1.7700	426.0	MAGHNIA		
AGM00060536		AG	34.8670	0.1500	752.0	SAIDA		
AGM00060555		AG	33.0680	6.0890	85.0	SIDI MAHDI		
AGM00060580		AG	31.9170	5.4130	150.0	OUARGLA		
AJ000037742		AJ	40.9000	47.3000	313.0	ORDJONIKIDZE,ZERNOSOVHOZ		
AJ000037747		AJ	40.6170	47.1500	15.0	EVLAKH AIRPORT		
AJ000037756		AJ	40.5330	48.9330	755.0	MARAZA		
AJ000037816		AJ	40.5000	46.1000	1655.0	DASHKESAN		
AJ000037899		AJ	39.7670	46.7500	1355.0	SHUSHA		
AJ000037914		AJ	39.9000	48.0000	-1.0	IMISLY		

20 rows × 21 columns

```
#  How many stations are there in total?
total_stations = stations_enriched.count()
print("total_stations: ", total_stations)

# How many stations were active so far in 2025
active_2025 = stations_enriched.filter(F.col("LASTYEAR_ANY") >= 2025).count()
print("active stations in 2025: ", active_2025)
```

```
total_stations:  129657
active stations in 2025:  38481
```

```
# station network belongings analysis
network_counts = stations_enriched.select(
    F.when(F.col("GSN_FLAG").contains("GSN"), 1).otherwise(0).alias("is_GSN"),
    F.when(F.col("HCN_CRN").contains("HCN"), 1).otherwise(0).alias("is_HCN"),
    F.when(F.col("HCN_CRN").contains("CRN"), 1).otherwise(0).alias("is_CRN"),
).agg(
    F.sum("is_GSN").alias("GSN"),
    F.sum("is_HCN").alias("HCN"),
    F.sum("is_CRN").alias("CRN"),
)

show_as_html(network_counts)
```

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

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

GSN	HCN	CRN
991	1218	234

```python
# Are there any stations that are in more than one of these networks?
# multi-network station counts query critera: net_count >= 2
flags = stations_enriched.select(
    "ID",
    F.when(F.col("GSN_FLAG").contains("GSN"), 1).otherwise(0).alias("is_GSN"),
    F.when(F.col("HCN_CRN").contains("HCN"), 1).otherwise(0).alias("is_HCN"),
    F.when(F.col("HCN_CRN").contains("CRN"), 1).otherwise(0).alias("is_CRN"),
)

multi_network = (
    flags.withColumn("net_count", F.col("is_GSN") + F.col("is_HCN") + F.col("is_CRN"))
    .filter(F.col("net_count") >= 2)
    .select("ID", "is_GSN", "is_HCN", "is_CRN", "net_count")
)
show_as_html(multi_network)

print(
    f"multi_network number is {multi_network.count()}, and they all overlap with GSN and HCN."
)
```

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

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

ID	is_GSN	is_HCN	is_CRN	net_count
USW00012921	1	1	0	2
USW00024128	1	1	0	2
USW00024213	1	1	0	2
USW00012836	1	1	0	2
USW00093193	1	1	0	2
USW00014771	1	1	0	2
USW00014922	1	1	0	2
USW00094008	1	1	0	2
USW00024144	1	1	0	2
USW00003870	1	1	0	2
USW00023044	1	1	0	2
USW00013782	1	1	0	2
USW00023051	1	1	0	2
USW00014742	1	1	0	2
USW00093729	1	1	0	2

```
multi_network number is 15, and they all overlap with GSN and HCN.
```

(b)

```python
# How many stations are there in the Southern_Hemisphere?
southern_hemisphere = stations_enriched.filter(F.col("LATITUDE") < 0).count()
print("southern_hemisphere count: ", southern_hemisphere)
```

```
southern_hemisphere count:  25357
```

```python
# identify us_territories, excluding the United States it self
# How many stations are there in total in the territories of the United States, excluding the United States itself?
us_territories = (
    stations_enriched.filter(
        (F.col("COUNTRY_NAME").contains("United States"))
        & (F.trim(F.col("COUNTRY_NAME")) != "United States")
```

```
    )
    .groupBy("COUNTRY_NAME")
    .agg(F.count("*").alias("STATION_NUM"))
)


show_as_html(us_territories)
us_territories_station_num = us_territories.agg(
    F.sum("station_num").alias("us_territories_station_num")
)
show_as_html(us_territories_station_num)
```

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

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

COUNTRY_NAME	STATION_NUM
Northern Mariana Islands [United States]	11
Puerto Rico [United States]	260
Guam [United States]	34
Johnston Atoll [United States]	4
American Samoa [United States]	21
Virgin Islands [United States]	77
Midway Islands [United States}	3
Palmyra Atoll [United States]	3
Wake Island [United States]	1

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

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

us_territories_station_num
414

(c)

```
# groupby country and states and count, save
station_num_by_country = (
    stations_enriched.groupBy("COUNTRY_CODE", "COUNTRY_NAME")
    .agg(F.count("*").alias("STATION_NUM"))
    .orderBy(F.desc("STATION_NUM"))
)

show_as_html(station_num_by_country)
```

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

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

COUNTRY_CODE	COUNTRY_NAME	STATION_NUM
US	United States	75846
AS	Australia	17088
CA	Canada	9269

BR	Brazil	5989
MX	Mexico	5249
IN	India	3807
SW	Sweden	1721
SF	South Africa	1166
RS	Russia	1123
GM	Germany	1123
FI	Finland	922
NO	Norway	461
NL	Netherlands	386
KZ	Kazakhstan	329
WA	Namibia	283
RQ	Puerto Rico [United States]	260
CH	China	228
SP	Spain	207
UP	Ukraine	204
JA	Japan	202

```
us_territories.write.parquet(station_count_us_terri_path, mode="overwrite")
station_count_by_country.write.parquet(station_count_by_country_path, mode="overwrite")
```

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

```
Found 2 items
-rw-r--r--   1 yxi75 supergroup          0 2025-09-11 22:33 wasbs://campus-user@madsstorage002.blob.core.windows.net/yxi75/station_count_us_te
-rw-r--r--   1 yxi75 supergroup      1.0 K 2025-09-11 22:33 wasbs://campus-user@madsstorage002.blob.core.windows.net/yxi75/station_count_us_te
Found 2 items
-rw-r--r--   1 yxi75 supergroup          0 2025-09-11 22:33 wasbs://campus-user@madsstorage002.blob.core.windows.net/yxi75/station_count_by_co
-rw-r--r--   1 yxi75 supergroup      4.9 K 2025-09-11 22:33 wasbs://campus-user@madsstorage002.blob.core.windows.net/yxi75/station_count_by_co
```

```
# load countries meta table for join
countries_df = (
    spark.read.text(paths["countries"])
    .withColumn("CODE", F.substring("value", 1, 2))
    .withColumn("COUNTRY_NAME", F.substring("value", 4, 61))
    .withColumnRenamed("CODE", "COUNTRY_CODE")
    .drop("value")
)

# join country counts onto countries
country_meta_with_station_num = countries_df.join(
    station_count_by_country.withColumnRenamed("count", "STATION_NUM"),
    on=["COUNTRY_NAME", "COUNTRY_CODE"],
    how="left",
).orderBy("COUNTRY_NAME")

show_as_html(country_meta_with_station_num)
```

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

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

COUNTRY_NAME	COUNTRY_CODE	STATION_NUM
Afghanistan	AF	4

Albania	AL	3
Algeria	AG	87
American Samoa [United States]	AQ	21
Angola	AO	6
Antarctica	AY	102
Antigua and Barbuda	AC	2
Argentina	AR	101
Armenia	AM	53
Australia	AS	17088
Austria	AU	13
Azerbaijan	AJ	66
Bahamas, The	BF	72
Bahrain	BA	1
Bangladesh	BG	10
Barbados	BB	1
Belarus	BO	51
Belgium	BE	1
Belize	BH	1
Benin	BN	9

```
# save to country_meta_with_station_num_path
country_meta_with_station_num.write.parquet(
    country_meta_with_station_num_path, "overwrite"
)
```

```
25/09/11 22:55:21 WARN AzureFileSystemThreadPoolExecutor: Disabling threads for Delete operation as thread count 0 is <= 1
```

```
!hdfs dfs -ls {country_meta_with_station_num_path}
```

```
Found 2 items
-rw-r--r--   1 yxi75 supergroup          0 2025-09-11 22:55 wasbs://campus-user@madsstorage002.blob.core.windows.net/yxi75/country_meta_with_s
-rw-r--r--   1 yxi75 supergroup       4945 2025-09-11 22:55 wasbs://campus-user@madsstorage002.blob.core.windows.net/yxi75/country_meta_with_s
```

```
station_count_by_state = stations_enriched.groupBy("STATE", "STATE_NAME").agg(
    F.count("*").alias("STATION_NUM")
)
show_as_html(station_count_by_state, 3)
```

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

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

STATE	STATE_NAME	STATION_NUM
NS	NOVA SCOTIA	398
NE	NEBRASKA	2436
IA	IOWA	1106

```
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, 3)
```

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

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

CODE	STATE_NAME
AB	ALBERTA
AK	ALASKA
AL	ALABAMA

```
states_meta_with_station_num = (
    states_df.withColumnRenamed("CODE", "STATE_CODE")
    .join(
        station_count_by_state.withColumnRenamed("STATE", "STATE_CODE"),
        on=["STATE_NAME", "STATE_CODE"],
        how="left",
    )
    .orderBy(F.desc("STATION_NUM"))
)

show_as_html(states_meta_with_station_num)
```

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

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

STATE_NAME	STATE_CODE	STATION_NUM
TEXAS	TX	6472
COLORADO	CO	4784
CALIFORNIA	CA	3166
NORTH CAROLINA	NC	2747
MINNESOTA	MN	2675
NEBRASKA	NE	2436
KANSAS	KS	2401
NEW MEXICO	NM	2295
FLORIDA	FL	2244
ILLINOIS	IL	2234
OREGON	OR	2031
ONTARIO	ON	2021
INDIANA	IN	2020
NEW YORK	NY	1912
TENNESSEE	TN	1755
BRITISH COLUMBIA	BC	1720
WASHINGTON	WA	1694

ARIZONA	AZ	1692
PENNSYLVANIA	PA	1641
MISSOURI	MO	1624

```
states_meta_with_station_num.write.parquet(
    states_meta_with_station_num_path, "overwrite"
)
```

```
25/09/11 23:18:14 WARN AzureFileSystemThreadPoolExecutor: Disabling threads for Delete operation as thread count 0 is <= 1
```

```
# check USER_ROOT storage status
!hdfs dfs -ls -h {USER_ROOT}
```

```
Found 5 items
drwxr-xr-x   - yxi75 supergroup          0 2025-09-11 22:55 wasbs://campus-user@madsstorage002.blob.core.windows.net/yxi75/country_meta_with_s
drwxr-xr-x   - yxi75 supergroup          0 2025-09-11 23:18 wasbs://campus-user@madsstorage002.blob.core.windows.net/yxi75/states_meta_with_st
drwxr-xr-x   - yxi75 supergroup          0 2025-09-11 22:33 wasbs://campus-user@madsstorage002.blob.core.windows.net/yxi75/station_count_by_co
drwxr-xr-x   - yxi75 supergroup          0 2025-09-11 22:33 wasbs://campus-user@madsstorage002.blob.core.windows.net/yxi75/station_count_us_te
drwxr-xr-x   - yxi75 supergroup          0 2025-09-11 18:30 wasbs://campus-user@madsstorage002.blob.core.windows.net/yxi75/stations_enriched_p
```

Q2 Distance User Define Function

(a) User Define Function: Haversine Distance

🌍 Haversine Distance Formula

The **Haversine formula** is used to calculate the great-circle distance between two points on a sphere (such as the Earth), given their latitude and longitude.

---

1. Definitions

Let two points be defined as:

- Point 1: (ϕ_1, λ_1)
- Point 2: (ϕ_2, λ_2)

where:

- ϕ = latitude (in radians)
- λ = longitude (in radians)
- R = radius of the Earth (mean radius ≈ 6,371 km)

---

2. Formula

The haversine function is defined as:

$$\text{hav}(\theta) = \sin^2\left(\frac{\theta}{2}\right) \tag{1}$$

Using this, the central angle c between two points is:

$$a = \sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1) \cdot \cos(\phi_2) \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right) \tag{2}$$

$$c = 2 \cdot \arcsin(\sqrt{a}) \tag{3}$$

Finally, the distance d is:

$$d = R \cdot c \tag{4}$$

where:

- $\Delta\phi = \phi_2 - \phi_1$
- $\Delta\lambda = \lambda_2 - \lambda_1$

---

3. Notes

- The Haversine formula accounts for Earth's curvature, making it more accurate than a simple Euclidean distance in geographic applications.
- It is sufficiently precise for most data science and GIS tasks.

- For centimeter-level accuracy (e.g., geodesy, GPS), ellipsoidal models such as **Vincenty's formula** or the **WGS84 geodesic** method should be used.

(This part is generated by chatGPT)

Haversine UDF

```
# Haversine UDF
from math import asin, cos, radians, sin, sqrt

from pyspark.sql.functions import udf


def haversine_km(lat1, lon1, lat2, lon2):
    R = 6371.0088  # mean Earth radius in km
    phi1, phi2 = radians(lat1), radians(lat2)
    dphi = radians(lat2 - lat1)
    dlambda = radians(lon2 - lon1)
    a = sin(dphi / 2) ** 2 + cos(phi1) * cos(phi2) * sin(dlambda / 2) ** 2
    c = 2 * asin(sqrt(a))
    return float(R * c)


# register UDF
haversine_udf = udf(haversine_km, DoubleType())
```

test udf within subset stations

```
# test udf within subset stations
test_stations = stations_enriched.limit(10)
show_as_html(test_stations)
```

```
.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	..
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			60545	..
AGM00060425		AG	36.2130	1.3320	141.1	ECH CHELIFF			60425	..
AGM00060430		AG	36.3000	2.2330	721.0	MILIANA			60430	..
AGM00060490		AG	35.6240	-0.6210	89.9	ES SENIA			60490	..
AGM00060514		AG	35.1670	2.3170	801.0	KSAR CHELLALA			60514	..
AGM00060515		AG	35.3330	4.2060	459.0	BOU SAADA			60515	..

10 rows × 21 columns

```
# generate station pairs

# SELF CROSS JOIN within test_stations
left = test_stations.select(
    F.col("ID").alias("ID_A"),
    F.col("NAME").alias("NAME_A"),
    F.col("LATITUDE").alias("LAT_A"),
    F.col("LONGITUDE").alias("LON_A"),
)
right = test_stations.select(
    F.col("ID").alias("ID_B"),
    F.col("NAME").alias("NAME_B"),
    F.col("LATITUDE").alias("LAT_B"),
    F.col("LONGITUDE").alias("LON_B"),
)
```

```

test_pairs = left.crossJoin(right).filter(F.col("ID_A") < F.col("ID_B"))

test_pairs = test_pairs.withColumn(
    "DIST_KM", haversine_udf("LAT_A", "LON_A", "LAT_B", "LON_B")
)
show_as_html(test_pairs)
```

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

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

ID_A	NAME_A	LAT_A	LON_A	ID_B	NAME_B	LAT_B	LON_B	DIST_KM
AG000060680	TAMANRASSET	22.800	5.4331	AGE00147794	BEJAIA-CAP CARBON	36.780	5.100	1554.836252
AG000060680	TAMANRASSET	22.800	5.4331	AGM00060351	JIJEL	36.795	5.874	1556.750841
AG000060680	TAMANRASSET	22.800	5.4331	AGM00060353	JIJEL-PORT	36.817	5.883	1559.219777
AG000060680	TAMANRASSET	22.800	5.4331	AGM00060419	MOHAMED BOUDIAF INTL	36.276	6.620	1502.817981
AG000060680	TAMANRASSET	22.800	5.4331	AGM00060550	EL-BAYADH	33.667	1.000	1283.611711
AG000060680	TAMANRASSET	22.800	5.4331	AGM00060563	HASSIR'MEL	32.933	3.283	1146.297735
AG000060680	TAMANRASSET	22.800	5.4331	AGM00060670	TISKA	24.293	9.452	442.002990
AG000060680	TAMANRASSET	22.800	5.4331	AJ000037639	AGSTAPHA AIRPORT	41.133	45.417	4236.617857
AG000060680	TAMANRASSET	22.800	5.4331	AJ000037656	ADJINAURSKAYA_STEP'	41.200	46.800	4350.094399
AGE00147794	BEJAIA-CAP CARBON	36.780	5.1000	AGM00060351	JIJEL	36.795	5.874	68.946174
AGE00147794	BEJAIA-CAP CARBON	36.780	5.1000	AGM00060353	JIJEL-PORT	36.817	5.883	69.838733
AGE00147794	BEJAIA-CAP CARBON	36.780	5.1000	AGM00060419	MOHAMED BOUDIAF INTL	36.276	6.620	146.921796
AGE00147794	BEJAIA-CAP CARBON	36.780	5.1000	AGM00060550	EL-BAYADH	33.667	1.000	508.348378
AGE00147794	BEJAIA-CAP CARBON	36.780	5.1000	AGM00060563	HASSIR'MEL	32.933	3.283	458.743957
AGE00147794	BEJAIA-CAP CARBON	36.780	5.1000	AGM00060670	TISKA	24.293	9.452	1449.209308
AGE00147794	BEJAIA-CAP CARBON	36.780	5.1000	AJ000037639	AGSTAPHA AIRPORT	41.133	45.417	3488.136797
AGE00147794	BEJAIA-CAP CARBON	36.780	5.1000	AJ000037656	ADJINAURSKAYA_STEP'	41.200	46.800	3602.576212
AGM00060351	JIJEL	36.795	5.8740	AGM00060353	JIJEL-PORT	36.817	5.883	2.574176
AGM00060351	JIJEL	36.795	5.8740	AGM00060419	MOHAMED BOUDIAF INTL	36.276	6.620	88.162740
AGM00060351	JIJEL	36.795	5.8740	AGM00060550	EL-BAYADH	33.667	1.000	562.845643

```
# select New Zealand stations
nz_stations = stations_enriched.filter(
    F.col("COUNTRY_NAME").contains("New Zealand")
).select("ID", "NAME", "LATITUDE", "LONGITUDE")

# SELF CROSS JOIN within newzealand
left = nz_stations.select(
    F.col("ID").alias("ID_A"),
    F.col("NAME").alias("NAME_A"),
    F.col("LATITUDE").alias("LAT_A"),
    F.col("LONGITUDE").alias("LON_A"),
)
right = nz_stations.select(
    F.col("ID").alias("ID_B"),
    F.col("NAME").alias("NAME_B"),
    F.col("LATITUDE").alias("LAT_B"),
    F.col("LONGITUDE").alias("LON_B"),
)

pairs = left.crossJoin(right).filter(F.col("ID_A") < F.col("ID_B"))

pairs = pairs.withColumn("DIST_KM", haversine_udf("LAT_A", "LON_A", "LAT_B", "LON_B"))
```

```
show_as_html(pairs)
```

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

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

ID_A	NAME_A	LAT_A	LON_A	ID_B	NAME_B	LAT_B	LON_B	DIST_KM
NZ000936150	HOKITIKA AERODROME	-42.717	170.983	NZ000937470	TARA HILLS	-44.517	169.900	218.309321
NZ000936150	HOKITIKA AERODROME	-42.717	170.983	NZ000939870	CHATHAM ISLANDS AWS	-43.950	-176.567	1015.251566
NZ000936150	HOKITIKA AERODROME	-42.717	170.983	NZM00093781	CHRISTCHURCH INTL	-43.489	172.532	152.258567
NZ000936150	HOKITIKA AERODROME	-42.717	170.983	NZM00093110	AUCKLAND AERO AWS	-37.000	174.800	714.127832
NZ000936150	HOKITIKA AERODROME	-42.717	170.983	TL000091724	NUKUNONO	-9.200	-171.917	4082.088199
NZ000936150	HOKITIKA AERODROME	-42.717	170.983	NZ000939450	CAMPBELL ISLAND AWS	-52.550	169.167	1101.720586
NZ000936150	HOKITIKA AERODROME	-42.717	170.983	NZM00093439	WELLINGTON AERO AWS	-41.333	174.800	350.796507
NZ000936150	HOKITIKA AERODROME	-42.717	170.983	NZM00093929	ENDERBY ISLAND AWS	-50.483	166.300	934.248833
NZ000936150	HOKITIKA AERODROME	-42.717	170.983	NZM00093678	KAIKOURA	-42.417	173.700	224.981589
NZ000937470	TARA HILLS	-44.517	169.900	NZ000939870	CHATHAM ISLANDS AWS	-43.950	-176.567	1078.799953
NZ000937470	TARA HILLS	-44.517	169.900	NZM00093781	CHRISTCHURCH INTL	-43.489	172.532	239.530461
NZ000939870	CHATHAM ISLANDS AWS	-43.950	-176.567	NZM00093781	CHRISTCHURCH INTL	-43.489	172.532	876.909075
NZ000937470	TARA HILLS	-44.517	169.900	NZM00093110	AUCKLAND AERO AWS	-37.000	174.800	931.744249
NZ000939870	CHATHAM ISLANDS AWS	-43.950	-176.567	NZM00093110	AUCKLAND AERO AWS	-37.000	174.800	1062.046480
NZ000937470	TARA HILLS	-44.517	169.900	TL000091724	NUKUNONO	-9.200	-171.917	4299.297371
NZ000937470	TARA HILLS	-44.517	169.900	NZ000939450	CAMPBELL ISLAND AWS	-52.550	169.167	894.846229
NZ000937470	TARA HILLS	-44.517	169.900	NZM00093439	WELLINGTON AERO AWS	-41.333	174.800	533.227413
NZ000937470	TARA HILLS	-44.517	169.900	NZM00093929	ENDERBY ISLAND AWS	-50.483	166.300	716.176359
NZ000939870	CHATHAM ISLANDS AWS	-43.950	-176.567	TL000091724	NUKUNONO	-9.200	-171.917	3890.098209
NZ000939870	CHATHAM ISLANDS AWS	-43.950	-176.567	NZM00093439	WELLINGTON AERO AWS	-41.333	174.800	763.267727

(b) closest station pair in NZ

```
closest_pair = pairs.orderBy(F.col("DIST_KM").asc()).limit(1)
closest_pair.show(truncate=False)

stationA = closest_pair.first()["ID_A"]
stationB = closest_pair.first()["ID_B"]
print(f"The closest station pair in New Zealand is station {stationA} and {stationB}")
```

```
+-----------+-----------------------------+-----+-------+-----------+-----------------------------+-------+-----+-----------------+
|ID_A       |NAME_A                       |LAT_A|LON_A  |ID_B       |NAME_B                       |LAT_B  |LON_B|DIST_KM          |
+-----------+-----------------------------+-----+-------+-----------+-----------------------------+-------+-----+-----------------+
|NZ000093417|PARAPARAUMU AWS              |-40.9|174.983|NZM00093439|WELLINGTON AERO AWS          |-41.333|174.8|50.52909627580285|
+-----------+-----------------------------+-----+-------+-----------+-----------------------------+-------+-----+-----------------+

The closest station pair in New Zealand is station NZ000093417 and NZM00093439
```

Q3 Daily Climate Summaries Study

(a) Core Element stat

```
# load daily
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
    ]
)

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

show_as_html(daily)
```

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

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

ID	DATE	ELEMENT	VALUE	MEASUREMENT	QUALITY	SOURCE	TIME
ASN00030019	20100101	PRCP	24.0	None	None	a	None
ASN00030021	20100101	PRCP	200.0	None	None	a	None
ASN00030022	20100101	TMAX	294.0	None	None	a	None
ASN00030022	20100101	TMIN	215.0	None	None	a	None
ASN00030022	20100101	PRCP	408.0	None	None	a	None
ASN00029121	20100101	PRCP	820.0	None	None	a	None
ASN00029126	20100101	TMAX	371.0	None	None	S	None
ASN00029126	20100101	TMIN	225.0	None	None	S	None
ASN00029126	20100101	PRCP	0.0	None	None	a	None
ASN00029126	20100101	TAVG	298.0	H	None	S	None
ASN00029127	20100101	TMAX	371.0	None	None	a	None
ASN00029127	20100101	TMIN	225.0	None	None	a	None
ASN00029127	20100101	PRCP	8.0	None	None	a	None
ASN00029129	20100101	PRCP	174.0	None	None	a	None
ASN00029130	20100101	PRCP	86.0	None	None	a	None
ASN00029131	20100101	PRCP	56.0	None	None	a	None
ASN00029132	20100101	PRCP	800.0	None	None	a	None
ASN00029136	20100101	PRCP	22.0	None	None	a	None
ASN00029137	20100101	PRCP	0.0	None	None	a	None
ASN00029139	20100101	TMAX	298.0	None	None	a	None

```
# filter daily records containing only the five core elements
core_elems = ["TMAX", "TMIN", "PRCP", "SNOW", "SNWD"]
daily_core = daily.filter(F.col("ELEMENT").isin(core_elems))
show_as_html(daily_core)
```

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

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

ID	DATE	ELEMENT	VALUE	MEASUREMENT	QUALITY	SOURCE	TIME
ASN00030019	20100101	PRCP	24.0	None	None	a	None

ASN00030021	20100101	PRCP	200.0	None	None	a	None
ASN00030022	20100101	TMAX	294.0	None	None	a	None
ASN00030022	20100101	TMIN	215.0	None	None	a	None
ASN00030022	20100101	PRCP	408.0	None	None	a	None
ASN00029121	20100101	PRCP	820.0	None	None	a	None
ASN00029126	20100101	TMAX	371.0	None	None	S	None
ASN00029126	20100101	TMIN	225.0	None	None	S	None
ASN00029126	20100101	PRCP	0.0	None	None	a	None
ASN00029127	20100101	TMAX	371.0	None	None	a	None
ASN00029127	20100101	TMIN	225.0	None	None	a	None
ASN00029127	20100101	PRCP	8.0	None	None	a	None
ASN00029129	20100101	PRCP	174.0	None	None	a	None
ASN00029130	20100101	PRCP	86.0	None	None	a	None
ASN00029131	20100101	PRCP	56.0	None	None	a	None
ASN00029132	20100101	PRCP	800.0	None	None	a	None
ASN00029136	20100101	PRCP	22.0	None	None	a	None
ASN00029137	20100101	PRCP	0.0	None	None	a	None
ASN00029139	20100101	TMAX	298.0	None	None	a	None
ASN00029139	20100101	TMIN	270.0	None	None	a	None

```
# groupby ELEMENT count
elem_counts = (
    daily_core.groupBy("ELEMENT")
    .agg(F.count("*").alias("OBSERVATION_COUNT"))
    .orderBy(F.desc("OBSERVATION_COUNT"))
)

show_as_html(elem_counts)
```

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

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

ELEMENT	OBSERVATION_COUNT
PRCP	1084610240
TMAX	461915395
TMIN	460752965
SNOW	361688529
SNWD	302055219

```
print(f"The most observations is PRCP.")
```

```
The most observations is PRCP.
```

(b) TMIN TMAX

```
# filter records which reports TMAX but TMIN: base on  (ID, DATE)
tmax = daily.filter(F.col("ELEMENT") == "TMAX").select(
    F.col("ID").alias("ID_T"), F.col("DATE").alias("DATE_T")
)

tmin = daily.filter(F.col("ELEMENT") == "TMIN").select(
    F.col("ID").alias("ID_N"), F.col("DATE").alias("DATE_N")
)
```

```
# left_anti join to find the part which left(tmax) have but right(tmin) haven't
tmax_no_tmin = tmax.join(
    tmin, (tmax.ID_T == tmin.ID_N) & (tmax.DATE_T == tmin.DATE_N), how="left_anti"
)
tmax_no_tmin_count = tmax_no_tmin.count()

show_as_html(tmax_no_tmin)
```

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

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

ID_T	DATE_T
AE000041196	19560704
AE000041196	19570607
AE000041196	19580617
AE000041196	19590525
AE000041196	19650714
AE000041196	19650813
AE000041196	19770612
AE000041196	19770904
AE000041196	19780117
AE000041196	19780619
AE000041196	19780910
AE000041196	19780918
AE000041196	19781018
AE000041196	19781024
AE000041196	19781113
AE000041196	19790215
AE000041196	19790420
AE000041196	19790508
AE000041196	19790517
AE000041196	19790609

```
print(f"Number of TMAX observations without corresponding TMIN: {tmax_no_tmin_count}")
```

```
Number of TMAX observations without corresponding TMIN: 10735252
```

```
# unique stations which contribute to missing_pairs count
unique_stations_missing = tmax_no_tmin.select("ID_T").distinct().count()
show_as_html(unique_stations_missing)

print(f"tmax_no_tmin_count = {tmax_no_tmin_count}")
print(f"unique_stations_missing = {unique_stations_missing}")
```

```
[Stage 233:==>           (16 + 8) / 107][Stage 234:>             (0 + 0) / 107]
```

```
print(f"Number of unique stations contributing to TMAX observations without corresponding TMIN: {unique_stations_missing}")
```

```
Number of unique stations contributing to TMAX observations without corresponding TMIN: 28751
```

```
stop_spark()
```

```
25/09/12 01:19:46 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

```

```