Ancy John

Student ID: 52770710

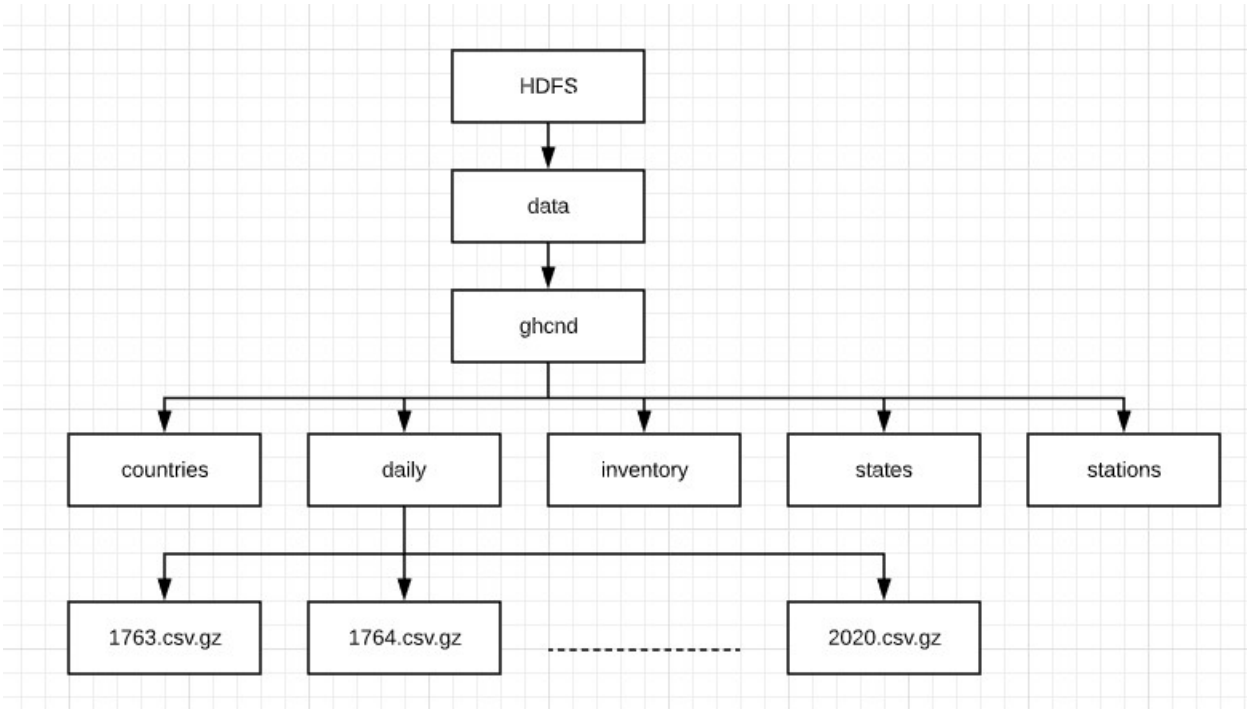
Q1. Define the separate data sources as daily, stations, states, countries, and inventory respectively. All of the data is stored in HDFS under hdfs:///data/ghcnd and is read only. Do not copy any of the data to your home directory.

Use the hdfs command to explore hdfs:///data/ghcnd without actually loading any data into memory.

(a) How is the data structured? Draw a directory tree to represent this in a sensible way.

The dataset 'ghcnd' is located in the directory 'data' in hdfs and consists of 4 different files and a subdirectory 'daily'. There are 258 files in the subdirectory 'daily'.

```
[ajo139@canterbury.ac.nz@mathmadslinux1p ~]$ hdfs dfs -ls /data/ghcnd
Found 5 items
-rwxr-xr-x   8 jsw93 jsw93       3659 2020-03-10 08:29 /data/ghcnd/countries
drwxr-xr-x   - jsw93 jsw93          0 2020-03-10 12:24 /data/ghcnd/daily
-rwxr-xr-x   8 jsw93 jsw93   31608486 2020-03-10 08:29 /data/ghcnd/inventory
-rwxr-xr-x   8 jsw93 jsw93       1086 2020-03-10 08:28 /data/ghcnd/states
-rwxr-xr-x   8 jsw93 jsw93    9896966 2020-03-10 08:28 /data/ghcnd/stations
[ajo139@canterbury.ac.nz@mathmadslinux1p ~]$ hdfs dfs -ls /data/ghcnd/countries
-rwxr-xr-x   8 jsw93 jsw93       3659 2020-03-10 08:29 /data/ghcnd/countries
```



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

Each file in daily represents each year. Thus, we have data from 258 different years, starting from 1763 until 2020. Size of the data starts from 3358 bytes in 1763, that almost doubles in 12 years. Thereafter we can see a steady increase in the data and reached on its peak in 2010 and

slightly got stabilized after that. Two months data for the year 2020 is also accessible from the directory.

```
[ajo139@canterbury.ac.nz@mathmadslinux1p ~]$ hdfs dfs -ls /data/ghcnd/daily
Found 258 items
-rw-r--r--   8 jsw93 jsw93       3358 2020-03-10 11:58 /data/ghcnd/daily/1763.csv.gz
-rw-r--r--   8 jsw93 jsw93       3327 2020-03-10 11:58 /data/ghcnd/daily/1764.csv.gz
-rw-r--r--   8 jsw93 jsw93       3335 2020-03-10 11:58 /data/ghcnd/daily/1765.csv.gz
-rw-r--r--   8 jsw93 jsw93       3344 2020-03-10 11:58 /data/ghcnd/daily/1766.csv.gz
```

(c) What is the total size of all of the data? How much of that is daily?

The actual size of the ghcnd data in bytes is 16680610588, and 16639100391 of those bytes is daily.

```
[ajo139@canterbury.ac.nz@mathmadslinux1p ~]$ hdfs dfs -du -s /data/ghcnd
16680610588  133444884704  /data/ghcnd
[ajo139@canterbury.ac.nz@mathmadslinux1p ~]$ hdfs dfs -du -s /data/ghcnd/daily
16639100391  133112803128  /data/ghcnd/daily
```

Q2. Start pyspark shell with 2 executors, 1 core per executor, 1 GB of executor memory, and 1 GB of master memory. You will now explore each data source briefly to ensure that the descriptions are accurate and that the data is as expected.

(a) Define schemas for each of daily, stations, states, countries, and inventory based on the descriptions in this assignment and the GHCN Daily README. These should use the data types defined in pyspark.sql.

```
spark = SparkSession.builder.getOrCreate()
sc = SparkContext.getOrCreate()

schema_daily = StructType([
    StructField("ID", StringType(), True),
    StructField("DATE", StringType(), True),
    StructField("ELEMENT", StringType(), True),
    StructField("VALUE", DoubleType(), True),
    StructField("MEASUREMENT FLAG", StringType(), True),
    StructField("QUALITY FLAG", StringType(), True),
    StructField("SOURCE FLAG", StringType(), True),
    StructField("OBSERVATION TIME", StringType(), True),
    ])

schema_stations = StructType([
    StructField("ID", StringType(), True),
    StructField("LATITUDE", DoubleType(), True),
    StructField("LONGITUDE", DoubleType(), True),
    StructField("ELEVATION", DoubleType(), True),
    StructField("STATE", StringType(), True),
    StructField("NAME", StringType(), True),
    StructField("GSN FLAG", StringType(), True),
    StructField("HCN/CRN FLAG", StringType(), True),
    StructField("WMO ID", StringType(), True),
    ])

schema_countries = StructType([
    StructField("CODE", StringType(), True),
    StructField("NAME", StringType(), True),
    ])

schema_states = StructType([
    StructField("CODE", StringType(), True),
    StructField("NAME", StringType(), True),
    ])

schema_inventory = StructType([
    StructField("ID", StringType(), True),
    StructField("LATITUDE", DoubleType(), True),
    StructField("LONGITUDE", DoubleType(), True),
    StructField("ELEMENT", StringType(), True),
    StructField("FIRSTYEAR", IntegerType(), True),
    StructField("LASTYEAR", IntegerType(), True),
    ])
```

Ancy John

Student ID: 52770710

(b) Load 1000 rows of hdfs:///data/ghcnd/daily/2020.csv.gz into Spark using the limit command immediately after the read command.

Was the description of the data accurate? Was there anything unexpected?

The description of the data is accurate enough. The 'ID' column which is expected to be a primary key is not unique.

```
In [7]: daily = (
   ...:     spark.read.format("com.databricks.spark.csv")
   ...:     .option("header", "false")
   ...:     .option("inferSchema", "false")
   ...:     .schema(schema_daily)
   ...:     .load("hdfs:///data/ghcnd/daily/2020.csv.gz")
   ...:     .limit(1000)
   ...: )

In [8]: daily.cache()
   ...: daily.show()
+-----------+--------+-------+-----+----------------+------------+-----------+----------------+
|         ID|    DATE|ELEMENT|VALUE|MEASUREMENT FLAG|QUALITY FLAG|SOURCE FLAG|OBSERVATION TIME|
+-----------+--------+-------+-----+----------------+------------+-----------+----------------+
|US1FLSL0019|20200101|   PRCP|  0.0|            null|        null|          N|            null|
|US1FLSL0019|20200101|   SNOW|  0.0|            null|        null|          N|            null|
|US1NVNY0012|20200101|   PRCP|  0.0|            null|        null|          N|            null|
|US1NVNY0012|20200101|   SNOW|  0.0|            null|        null|          N|            null|
|US1ILWM0012|20200101|   PRCP|  0.0|            null|        null|          N|            null|
|USS0018D08S|20200101|   TMAX| 46.0|            null|        null|          T|            null|
|USS0018D08S|20200101|   TMIN|  6.0|            null|        null|          T|            null|
|USS0018D08S|20200101|   TOBS|  6.0|            null|        null|          T|            null|
|USS0018D08S|20200101|   PRCP| 76.0|            null|        null|          T|            null|
|USS0018D08S|20200101|   SNWD|  0.0|            null|        null|          T|            null|
|USS0018D08S|20200101|   TAVG| 23.0|            null|        null|          T|            null|
|USS0018D08S|20200101|   WESD|127.0|            null|        null|          T|            null|
|USC00141761|20200101|   TMAX| 39.0|            null|        null|          7|            0700|
|USC00141761|20200101|   TMIN|-50.0|            null|        null|          7|            0700|
|USC00141761|20200101|   TOBS|-17.0|            null|        null|          7|            0700|
|USC00141761|20200101|   PRCP|  0.0|            null|        null|          7|            0700|
|USC00141761|20200101|   SNOW|  0.0|            null|        null|          7|            null|
|USC00141761|20200101|   SNWD|  0.0|            null|        null|          7|            0700|
|USC00141761|20200101|   WESD|  0.0|            null|        null|          7|            null|
|RQC00660061|20200101|   TMAX|289.0|            null|        null|          7|            0800|
+-----------+--------+-------+-----+----------------+------------+-----------+----------------+
only showing top 20 rows
```

(**c) Load each of stations, states, countries, and inventory into Spark as well. You will need to find a way to parse the fixed width text formatting, as this format is not included in the standard spark.read library. You could try using spark.read.format('text') and pyspark.sql.functions.substring or finding an existing open source library.**

How many rows are in each metadata table? How many stations do not have a WMO ID?

Ancy John

Student ID: 52770710

```
In [3]: stations = (
   ...:     spark.read.format("text")
   ...:     .load("hdfs:///data/ghcnd/stations")
   ...: )

In [4]: stations.select(
   ...:     F.trim(F.substring(F.col('value'),1,11)).alias('ID').cast(schema_stations['ID'].dataType),
   ...:     F.trim(F.substring(F.col('value'),13,8)).alias('LATITUDE').cast(schema_stations['LATITUDE'].dataType),
   ...:     F.trim(F.substring(F.col('value'),22,9)).alias('LONGITUDE').cast(schema_stations['LONGITUDE'].dataType),
   ...:     F.trim(F.substring(F.col('value'),32,6)).alias('ELEVATION').cast(schema_stations['ELEVATION'].dataType),
   ...:     F.trim(F.substring(F.col('value'),39,2)).alias('STATE').cast(schema_stations['STATE'].dataType),
   ...:     F.trim(F.substring(F.col('value'),42,30)).alias('NAME').cast(schema_stations['NAME'].dataType),
   ...:     F.trim(F.substring(F.col('value'),73,3)).alias('GSN_FLAG').cast(schema_stations['GSN_FLAG'].dataType),
   ...:     F.trim(F.substring(F.col('value'),77,3)).alias('CRN_FLAG').cast(schema_stations['HCN_FLAG'].dataType),
   ...:     F.trim(F.substring(F.col('value'),81,5)).alias('WMO_ID').cast(schema_stations['WMO_ID'].dataType),
   ...: ).show()
+-----------+--------+---------+---------+-----+--------------------+--------+--------+------+
|         ID|LATITUDE|LONGITUDE|ELEVATION|STATE|                NAME|GSN_FLAG|CRN_FLAG|WMO_ID|
+-----------+--------+---------+---------+-----+--------------------+--------+--------+------+
|ACW00011604| 17.1167| -61.7833|     10.1|     |ST JOHNS COOLIDGE...|        |        |      |
|ACW00011647| 17.1333| -61.7833|     19.2|     |            ST JOHNS|        |        |      |
|AE000041196|  25.333|   55.517|     34.0|     | SHARJAH INTER. AIRP|     GSN|        | 41196|
|AEM00041194|  25.255|   55.364|     10.4|     |          DUBAI INTL|        |        | 41194|
|AEM00041217|  24.433|   54.651|     26.8|     |      ABU DHABI INTL|        |        | 41217|
|AEM00041218|  24.262|   55.609|    264.9|     |         AL AIN INTL|        |        | 41218|
|AF000040930|  35.317|   69.017|   3366.0|     |        NORTH-SALANG|     GSN|        | 40930|
|AFM00040938|   34.21|   62.228|    977.2|     |               HERAT|        |        | 40938|
|AFM00040948|  34.566|   69.212|   1791.3|     |          KABUL INTL|        |        | 40948|
|AFM00040990|    31.5|    65.85|   1010.0|     |    KANDAHAR AIRPORT|        |        | 40990|
|AG000060390| 36.7167|     3.25|     24.0|     |  ALGER-DAR EL BEIDA|     GSN|        | 60390|
|AG000060590| 30.5667|   2.8667|    397.0|     |            EL-GOLEA|     GSN|        | 60590|
|AG000060611|   28.05|   9.6331|    561.0|     |           IN-AMENAS|     GSN|        | 60611|
|AG000060680|    22.8|   5.4331|   1362.0|     |         TAMANRASSET|     GSN|        | 60680|
|AGE00135039| 35.7297|     0.65|     50.0|     |ORAN-HOPITAL MILI...|        |        |      |
|AGE00147704|   36.97|     7.79|    161.0|     | ANNABA-CAP DE GARDE|        |        |      |
|AGE00147705|   36.78|     3.07|     59.0|     |ALGIERS-VILLE/UNI...|        |        |      |
|AGE00147706|    36.8|     3.03|    344.0|     |   ALGIERS-BOUZAREAH|        |        |      |
|AGE00147707|    36.8|     3.04|     38.0|     |  ALGIERS-CAP CAXINE|        |        |      |
|AGE00147708|   36.72|     4.05|    222.0|     |          TIZI OUZOU|        |        | 60395|
+-----------+--------+---------+---------+-----+--------------------+--------+--------+------+
only showing top 20 rows
```

```
   ...:
   ...: states = (
   ...:     spark.read.format("text")
   ...:     .load("hdfs:///data/ghcnd/states")
   ...: )
   ...:
   ...: states_ps = states.select(
   ...:     F.trim(F.substring(F.col('value'),1,2)).alias('CODE').cast(schema_states['CODE'].dataType),
   ...:     F.trim(F.substring(F.col('value'),4,47)).alias('NAME').cast(schema_states['NAME'].dataType),
   ...: )

In [5]: states_ps.cache()
   ...: states_ps.show()
+----+--------------------+
|CODE|                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|
+----+--------------------+
only showing top 20 rows
```

Ancy John

Student ID: 52770710

```
In [9]: countries = (
   ...:     spark.read.format("text")
   ...:     .load("hdfs:///data/ghcnd/countries")
   ...: )

In [10]: schema_countries = StructType([
    ...:     StructField("CODE", StringType(), True),
    ...:     StructField("NAME", StringType(), True),
    ...:     ])
    ...:
    ...: countries = (
    ...:     spark.read.format("text")
    ...:     .load("hdfs:///data/ghcnd/countries")
    ...: )
    ...:
    ...: countries.select(
    ...:     F.trim(F.substring(F.col('value'),1,2)).alias('CODE').cast(schema_countries['CODE'].dataType),
    ...:     F.trim(F.substring(F.col('value'),4,47)).alias('NAME').cast(schema_countries['NAME'].dataType),
    ...: ).show()
+----+--------------------+
|CODE|                NAME|
+----+--------------------+
|  AC| Antigua and Barbuda|
|  AE|United Arab Emirates|
|  AF|         Afghanistan|
|  AG|             Algeria|
|  AJ|          Azerbaijan|
|  AL|             Albania|
|  AM|             Armenia|
|  AO|              Angola|
|  AQ|American Samoa [U...|
|  AR|           Argentina|
|  AS|           Australia|
|  AU|             Austria|
|  AY|          Antarctica|
|  BA|             Bahrain|
|  BB|            Barbados|
|  BC|            Botswana|
|  BD|Bermuda [United K...|
|  BE|             Belgium|
|  BF|        Bahamas, The|
|  BG|          Bangladesh|
+----+--------------------+
only showing top 20 rows
```

```
In [14]: schema_inventory = StructType([
    ...:     StructField("ID", StringType(), True),
    ...:     StructField("LATITUDE", FloatType(), True),
    ...:     StructField("LONGITUDE", FloatType(), True),
    ...:     StructField("ELEMENT", StringType(), True),
    ...:     StructField("FIRSTYEAR", IntegerType(), True),
    ...:     StructField("LASTYEAR", IntegerType(), True),
    ...:     ])
    ...:
    ...:
    ...: inventory = (
    ...:     spark.read.format("text")
    ...:     .load("hdfs:///data/ghcnd/inventory")
    ...: )
    ...:
    ...: inventory.select(
    ...:     F.trim(F.substring(F.col('value'),1,11)).alias('ID').cast(schema_inventory['ID'].dataType),
    ...:     F.trim(F.substring(F.col('value'),13,8)).alias('LATITUDE').cast(schema_inventory['LATITUDE'].dataType),
    ...:     F.trim(F.substring(F.col('value'),22,9)).alias('LONGITUDE').cast(schema_inventory['LONGITUDE'].dataType),
    ...:     F.trim(F.substring(F.col('value'),32,4)).alias('ELEMENT').cast(schema_inventory['ELEMENT'].dataType),
    ...:     F.trim(F.substring(F.col('value'),37,4)).alias('FIRSTYEAR').cast(schema_inventory['FIRSTYEAR'].dataType),
    ...:     F.trim(F.substring(F.col('value'),42,4)).alias('LASTYEAR').cast(schema_inventory['LASTYEAR'].dataType),
    ...: ).show()
+-----------+--------+---------+-------+---------+--------+
|         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|  25.333|   55.517|   TMAX|     1944|    2019|
|AE000041196|  25.333|   55.517|   TMIN|     1944|    2020|
+-----------+--------+---------+-------+---------+--------+
only showing top 20 rows
```

Ancy John

Student ID: 52770710

Number of rows in each metadata table is:

```
In [35]: inventory_ps.count()
Out[35]: 687141

In [36]: states_ps.count()
Out[36]: 74

In [37]: countries_ps.count()
Out[37]: 219

In [38]: stations_ps.count()
Out[38]: 115081
```

The number of station without a 'WMO_ID'

```
In [46]: counts = (
    ...:     stations_ps
    ...:     # Select only the columns that are needed
    ...:     .select(['ID', 'WMO_ID'])
    ...:     .filter('WMO_ID = ""')
    ...: )
    ...: counts.count()
Out[46]: 106993
```

Q3. Next you will combine the daily climate summaries with the metadata tables, joining on station, state, and country. Note that joining the daily climate summaries and metadata into a single table is not efficient for a database of this size but joining the metadata into a single table is very convenient for filtering and sorting based on attributes at a station level.

You will need to start saving some intermediate outputs along the way. Create an output directory in your home directory (e.g. hdfs:///user/abc123/outputs/ghcnd/). Note that we only have 400GB of storage available in total, so think carefully before you write output to HDFS.

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

Ancy John

Student ID: 52770710

```
In [29]:  df = stations_ps.withColumn('CODE', F.substring(stations_ps['ID'],1,2))

In [30]: df.show()
+-----------+--------+---------+---------+-----+--------------------+--------+--------+------+----+
|         ID|LATITUDE|LONGITUDE|ELEVATION|STATE|                NAME|GSN_FLAG|CRN_FLAG|WMO_ID|CODE|
+-----------+--------+---------+---------+-----+--------------------+--------+--------+------+----+
|ACW00011604| 17.1167| -61.7833|     10.1|     |ST JOHNS COOLIDGE...|        |        |      |  AC|
|ACW00011647| 17.1333| -61.7833|     19.2|     |            ST JOHNS|        |        |      |  AC|
|AE000041196|  25.333|   55.517|     34.0|     | SHARJAH INTER. AIRP|     GSN|        | 41196|  AE|
|AEM00041194|  25.255|   55.364|     10.4|     |          DUBAI INTL|        |        | 41194|  AE|
|AEM00041217|  24.433|   54.651|     26.8|     |       ABU DHABI INTL|        |        | 41217|  AE|
|AEM00041218|  24.262|   55.609|    264.9|     |          AL AIN INTL|        |        | 41218|  AE|
|AF000040930|  35.317|   69.017|   3366.0|     |         NORTH-SALANG|     GSN|        | 40930|  AF|
|AFM00040938|   34.21|   62.228|    977.2|     |               HERAT|        |        | 40938|  AF|
|AFM00040948|  34.566|   69.212|   1791.3|     |           KABUL INTL|        |        | 40948|  AF|
|AFM00040990|    31.5|    65.85|   1010.0|     |     KANDAHAR AIRPORT|        |        | 40990|  AF|
|AG000060390| 36.7167|     3.25|     24.0|     |   ALGER-DAR EL BEIDA|     GSN|        | 60390|  AG|
|AG000060590| 30.5667|   2.8667|    397.0|     |             EL-GOLEA|     GSN|        | 60590|  AG|
|AG000060611|   28.05|   9.6331|    561.0|     |            IN-AMENAS|     GSN|        | 60611|  AG|
|AG000060680|    22.8|   5.4331|   1362.0|     |          TAMANRASSET|     GSN|        | 60680|  AG|
|AGE00135039| 35.7297|     0.65|     50.0|     |ORAN-HOPITAL MILI...|        |        |      |  AG|
|AGE00147704|   36.97|     7.79|    161.0|     | ANNABA-CAP DE GARDE|        |        |      |  AG|
|AGE00147705|   36.78|     3.07|     59.0|     |ALGIERS-VILLE/UNI...|        |        |      |  AG|
|AGE00147706|    36.8|     3.03|    344.0|     |    ALGIERS-BOUZAREAH|        |        |      |  AG|
|AGE00147707|    36.8|     3.04|     38.0|     |   ALGIERS-CAP CAXINE|        |        |      |  AG|
|AGE00147708|   36.72|     4.05|    222.0|     |            TIZI OUZOU|        |        | 60395|  AG|
+-----------+--------+---------+---------+-----+--------------------+--------+--------+------+----+
only showing top 20 rows
```

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

```
In [39]: df1 = df1.withColumnRenamed('country_name', 'COUNTRY')

In [40]: df1.show()
+----+-----------+--------+---------+---------+-----+--------------------+--------+--------+------+--------------------+
|CODE|         ID|LATITUDE|LONGITUDE|ELEVATION|STATE|                NAME|GSN_FLAG|CRN_FLAG|WMO_ID|             COUNTRY|
+----+-----------+--------+---------+---------+-----+--------------------+--------+--------+------+--------------------+
|  AC|ACW00011604| 17.1167| -61.7833|     10.1|     |ST JOHNS COOLIDGE...|        |        |      | Antigua and Barbuda|
|  AC|ACW00011647| 17.1333| -61.7833|     19.2|     |            ST JOHNS|        |        |      | Antigua and Barbuda|
|  AE|AE000041196|  25.333|   55.517|     34.0|     | SHARJAH INTER. AIRP|     GSN|        | 41196|United Arab Emirates|
|  AE|AEM00041194|  25.255|   55.364|     10.4|     |          DUBAI INTL|        |        | 41194|United Arab Emirates|
|  AE|AEM00041217|  24.433|   54.651|     26.8|     |       ABU DHABI INTL|        |        | 41217|United Arab Emirates|
|  AE|AEM00041218|  24.262|   55.609|    264.9|     |          AL AIN INTL|        |        | 41218|United Arab Emirates|
|  AF|AF000040930|  35.317|   69.017|   3366.0|     |         NORTH-SALANG|     GSN|        | 40930|         Afghanistan|
|  AF|AFM00040938|   34.21|   62.228|    977.2|     |               HERAT|        |        | 40938|         Afghanistan|
|  AF|AFM00040948|  34.566|   69.212|   1791.3|     |           KABUL INTL|        |        | 40948|         Afghanistan|
|  AF|AFM00040990|    31.5|    65.85|   1010.0|     |     KANDAHAR AIRPORT|        |        | 40990|         Afghanistan|
|  AG|AG000060390| 36.7167|     3.25|     24.0|     |   ALGER-DAR EL BEIDA|     GSN|        | 60390|             Algeria|
|  AG|AG000060590| 30.5667|   2.8667|    397.0|     |             EL-GOLEA|     GSN|        | 60590|             Algeria|
|  AG|AG000060611|   28.05|   9.6331|    561.0|     |            IN-AMENAS|     GSN|        | 60611|             Algeria|
|  AG|AG000060680|    22.8|   5.4331|   1362.0|     |          TAMANRASSET|     GSN|        | 60680|             Algeria|
|  AG|AGE00135039| 35.7297|     0.65|     50.0|     |ORAN-HOPITAL MILI...|        |        |      |             Algeria|
|  AG|AGE00147704|   36.97|     7.79|    161.0|     | ANNABA-CAP DE GARDE|        |        |      |             Algeria|
|  AG|AGE00147705|   36.78|     3.07|     59.0|     |ALGIERS-VILLE/UNI...|        |        |      |             Algeria|
|  AG|AGE00147706|    36.8|     3.03|    344.0|     |    ALGIERS-BOUZAREAH|        |        |      |             Algeria|
|  AG|AGE00147707|    36.8|     3.04|     38.0|     |   ALGIERS-CAP CAXINE|        |        |      |             Algeria|
|  AG|AGE00147708|   36.72|     4.05|    222.0|     |            TIZI OUZOU|        |        | 60395|             Algeria|
+----+-----------+--------+---------+---------+-----+--------------------+--------+--------+------+--------------------+
only showing top 20 rows
```

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

Ancy John

Student ID: 52770710

```
In [20]: df3 = df2.na.fill({'STATE': ''})

In [21]: df3.show()
+----------+------------+-----------+--------+---------+---------+--------------------+--------+--------+------+--------------------+-----+
|STATE_CODE|COUNTRY_CODE|         ID|LATITUDE|LONGITUDE|ELEVATION|                NAME|GSN_FLAG|CRN_FLAG|WMO_ID|             COUNTRY|STATE|
+----------+------------+-----------+--------+---------+---------+--------------------+--------+--------+------+--------------------+-----+
|          |          AC|ACW00011604| 17.1167| -61.7833|     10.1|ST JOHNS COOLIDGE...|        |        |      | Antigua and Barbuda|     |
|          |          AC|ACW00011647| 17.1333| -61.7833|     19.2|            ST JOHNS|        |        |      | Antigua and Barbuda|     |
|          |          AE|AE000041196|  25.333|   55.517|     34.0| SHARJAH INTER. AIRP|     GSN|        | 41196|United Arab Emirates|     |
|          |          AE|AEM00041194|  25.255|   55.364|     10.4|          DUBAI INTL|        |        | 41194|United Arab Emirates|     |
|          |          AE|AEM00041217|  24.433|   54.651|     26.8|      ABU DHABI INTL|        |        | 41217|United Arab Emirates|     |
|          |          AE|AEM00041218|  24.262|   55.609|     264.9|         AL AIN INTL|        |        | 41218|United Arab Emirates|     |
|          |          AF|AF000040930|  35.317|   69.017|   3366.0|        NORTH-SALANG|     GSN|        | 40930|         Afghanistan|     |
|          |          AF|AFM00040938|   34.21|   62.228|    977.2|               HERAT|        |        | 40938|         Afghanistan|     |
|          |          AF|AFM00040948|  34.566|   69.212|   1791.3|          KABUL INTL|        |        | 40948|         Afghanistan|     |
|          |          AF|AFM00040990|    31.5|    65.85|   1010.0|    KANDAHAR AIRPORT|        |        | 40990|         Afghanistan|     |
|          |          AG|AG000060390| 36.7167|     3.25|     24.0|  ALGER-DAR EL BEIDA|     GSN|        | 60390|             Algeria|     |
|          |          AG|AG000060590| 30.5667|   2.8667|    397.0|           EL-GOLEA|     GSN|        | 60590|             Algeria|     |
|          |          AG|AG000060611|   28.05|   9.6331|    561.0|          IN-AMENAS|     GSN|        | 60611|             Algeria|     |
|          |          AG|AG000060680|    22.8|   5.4331|   1362.0|        TAMANRASSET|     GSN|        | 60680|             Algeria|     |
|          |          AG|AGE00135039| 35.7297|     0.65|     50.0|ORAN-HOPITAL MILI...|        |        |      |             Algeria|     |
|          |          AG|AGE00147704|   36.97|     7.79|    161.0| ANNABA-CAP DE GARDE|        |        |      |             Algeria|     |
|          |          AG|AGE00147705|   36.78|     3.07|     59.0|ALGIERS-VILLE/UNI...|        |        |      |             Algeria|     |
|          |          AG|AGE00147706|    36.8|     3.03|    344.0|   ALGIERS-BOUZAREAH|        |        |      |             Algeria|     |
|          |          AG|AGE00147707|    36.8|     3.04|     38.0|  ALGIERS-CAP CAXINE|        |        |      |             Algeria|     |
|          |          AG|AGE00147708|   36.72|     4.05|    222.0|          TIZI OUZOU|        |        | 60395|             Algeria|     |
+----------+------------+-----------+--------+---------+---------+--------------------+--------+--------+------+--------------------+-----+
only showing top 20 rows
```

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

```
In [17]: Range = (
    ...:     inventory_ps
    ...:     # Select only the columns that are needed
    ...:     .select(['ID', 'FIRSTYEAR', 'LASTYEAR'])
    ...:     # Group by ID and find minimum and maximum
    ...:     .groupBy('ID')
    ...:     .agg({'FIRSTYEAR': 'min', 'LASTYEAR': 'max'})
    ...:     #.agg({'LASTYEAR': 'max'})
    ...:     .select(
    ...:         F.col('ID').alias('STATION_ID'),
    ...:         #F.col('ELEMENT'),
    ...:         F.col('min(FIRSTYEAR)').alias('FIRSTYEAR'),
    ...:         F.col('max(LASTYEAR)').alias('LASTYEAR')
    ...:     )
    ...: )

In [18]: Range.show()
+-----------+---------+--------+
| STATION_ID|FIRSTYEAR|LASTYEAR|
+-----------+---------+--------+
|ACW00011647|     1957|    1970|
|AEM00041217|     1983|    2020|
|AG000060590|     1892|    2020|
|AGE00147706|     1893|    1920|
|AGE00147708|     1879|    2020|
|AGE00147709|     1879|    1938|
|AGE00147710|     1909|    2009|
|AGE00147711|     1880|    1938|
|AGE00147714|     1896|    1938|
|AGE00147719|     1888|    2020|
|AGM00060351|     1981|    2020|
|AGM00060353|     1996|    2019|
|AGM00060360|     1945|    2020|
|AGM00060387|     1995|    2004|
|AGM00060445|     1957|    2020|
|AGM00060452|     1985|    2020|
|AGM00060467|     1981|    2019|
|AGM00060468|     1973|    2020|
|AGM00060507|     1943|    2020|
|AGM00060511|     1983|    2020|
+-----------+---------+--------+
only showing top 20 rows
```

Ancy John

Student ID: 52770710

How many different elements has each station collected overall?

```
In [19]: counts = (
    ...:     inventory_ps
    ...:     # Select only the columns that are needed
    ...:     .select(['ID', 'ELEMENT'])
    ...:     # Group by ID and find minimum and maximum
    ...:     .groupBy('ID')
    ...:     .agg({'ELEMENT': 'count'})
    ...:     .select(
    ...:         F.col('ID').alias('STATION_ID'),
    ...:         F.col('count(ELEMENT)').alias('ELEMENT_COUNT')
    ...:     )
    ...: )

In [20]: counts.show()
+-----------+-------------+
| STATION_ID|ELEMENT_COUNT|
+-----------+-------------+
|ACW00011647|            7|
|AEM00041217|            4|
|AG000060590|            4|
|AGE00147706|            3|
|AGE00147708|            5|
|AGE00147709|            3|
|AGE00147710|            4|
|AGE00147711|            3|
|AGE00147714|            3|
|AGE00147719|            4|
|AGM00060351|            4|
|AGM00060353|            4|
|AGM00060360|            4|
|AGM00060387|            4|
|AGM00060445|            5|
|AGM00060452|            4|
|AGM00060467|            4|
|AGM00060468|            5|
|AGM00060507|            5|
|AGM00060511|            5|
+-----------+-------------+
only showing top 20 rows
```

Further, count separately the number of core elements and the number of ”other” elements that each station has collected overall.

```
In [35]: other.show()
+-----------+-------------+--------------+
| STATION_ID|CORE ELEMENTS|OTHER ELEMENTS|
+-----------+-------------+--------------+
|ACW00011647|            5|             2|
|AEM00041217|            3|             1|
|AG000060590|            3|             1|
|AGE00147708|            4|             1|
|AGE00147710|            3|             1|
|AGE00147719|            3|             1|
|AGM00060351|            3|             1|
|AGM00060353|            3|             1|
|AGM00060360|            3|             1|
|AGM00060387|            3|             1|
|AGM00060445|            4|             1|
|AGM00060452|            3|             1|
|AGM00060467|            3|             1|
|AGM00060468|            4|             1|
|AGM00060507|            4|             1|
|AGM00060511|            4|             1|
|AGM00060535|            4|             1|
|AGM00060540|            4|             1|
|AGM00060602|            4|             1|
|AGM00060603|            3|             1|
+-----------+-------------+--------------+
only showing top 20 rows
```

Ancy John

Student ID: 52770710

How many stations collect all five core elements?

Only one station collects all the 5 core elements.

How many only collected temperature?

I have used the array_contains() method tho check for the other core elements and eliminated those rows, by keeping the rows with temperature elements only. 1864 stations collected only temperature.

```
In [103]: core2 = core1.withColumn('temperature_only', array_contains(core1.CORE_ELEMENTS, 'SNOW'))

In [104]: core2 = core2.filter("temperature_only == False")

In [105]: core2.count()
Out[105]: 1898

In [106]: core2 = core2.drop('temperature_only')

In [107]: core3 = core2.withColumn('temperature_only', array_contains(core2.CORE_ELEMENTS, 'SNWD'))

In [108]: core3 = core3.filter("temperature_only == False")

In [109]: core3.count()
Out[109]: 1864
```

(e) LEFT JOIN stations and your output from part (d).

```
In [118]: stations_enriched = stations_enriched.join(other1, on=['ID'], how='left')

In [119]: stations_enriched.show()
+-----------+----------+------------+--------+---------+---------+--------------------+--------+--------+------+--------------------+-----+---------+--------------------+----------+---------+----------+
|         ID|STATE_CODE|COUNTRY_CODE|LATITUDE|LONGITUDE|ELEVATION|                NAME|GSN_FLAG|CRN_FLAG|WMO_ID|             COUNTRY|STATE|CORE_TYPE|        CORE_ELEMENTS|OTHER_TYPE|FIRSTYEAR|LASTSTYEAR|
+-----------+----------+------------+--------+---------+---------+--------------------+--------+--------+------+--------------------+-----+---------+--------------------+----------+---------+----------+
|ACW00011647|          |          AC| 17.1333| -61.7833|     19.2|            ST JOHNS|        |        |      | Antigua and Barbuda|     |        5|[TMAX, TMIN, PRCP...|         2|     1957|      1970|
|AEM00041217|          |          AE|  24.433|   54.651|     26.8|       ABU DHABI INTL|        |        | 41217|United Arab Emirates|     |        3|  [TMAX, TMIN, PRCP]|         1|     1983|      2020|
|AG000060590|          |          AG| 30.5667|   2.8667|    397.0|            EL-GOLEA|     GSN|        | 60590|             Algeria|     |        3|  [TMAX, TMIN, PRCP]|         1|     1892|      2020|
|AGE00147706|          |          AG|    36.8|     3.03|    344.0|   ALGIERS-BOUZAREAH|        |        |      |             Algeria|     |     null|                null|      null|     null|      null|
|AGE00147708|          |          AG|   36.72|     4.05|    222.0|          TIZI OUZOU|        |        | 60395|             Algeria|     |        4|[TMAX, TMIN, PRCP...|         1|     1879|      2020|
|AGE00147709|          |          AG|   36.63|      4.2|    942.0|        FORT NATIONAL|        |        |      |             Algeria|     |     null|                null|      null|     null|      null|
|AGE00147710|          |          AG|   36.75|      5.1|      9.0|BEJAIA-BOUGIE (PORT)|        |        | 60401|             Algeria|     |        3|  [TMAX, TMIN, PRCP]|         1|     1909|      2009|
|AGE00147711|          |          AG| 36.3697|     6.62|    660.0|          CONSTANTINE|        |        |      |             Algeria|     |     null|                null|      null|     null|      null|
|AGE00147714|          |          AG|   35.77|      0.8|     78.0|      ORAN-CAP FALCON|        |        |      |             Algeria|     |     null|                null|      null|     null|      null|
|AGE00147719|          |          AG| 33.7997|     2.89|    767.0|             LAGHOUAT|        |        | 60545|             Algeria|     |        3|  [TMAX, TMIN, PRCP]|         1|     1888|      2020|
|AGM00060351|          |          AG|  36.795|    5.874|     11.0|               JIJEL|        |        | 60351|             Algeria|     |        3|  [TMAX, TMIN, PRCP]|         1|     1981|      2020|
|AGM00060353|          |          AG|  36.817|    5.883|      6.0|          JIJEL-PORT|        |        | 60353|             Algeria|     |        3|  [TMAX, TMIN, PRCP]|         1|     1996|      2019|
|AGM00060360|          |          AG|  36.822|    7.809|      4.9|              ANNABA|        |        | 60360|             Algeria|     |        3|  [TMAX, TMIN, PRCP]|         1|     1945|      2020|
|AGM00060387|          |          AG|  36.917|     3.95|      8.0|              DELLYS|        |        | 60387|             Algeria|     |        3|  [TMAX, TMIN, PRCP]|         1|     1995|      2004|
|AGM00060445|          |          AG|  36.178|    5.324|   1050.0|     SETIF AIN ARNAT|        |        | 60445|             Algeria|     |        4|[TMAX, TMIN, PRCP...|         1|     1957|      2020|
|AGM00060452|          |          AG|  35.817|   -0.267|      4.0|               ARZEW|        |        | 60452|             Algeria|     |        3|  [TMAX, TMIN, PRCP]|         1|     1985|      2020|
|AGM00060467|          |          AG|  35.667|      4.5|    442.0|            M'SILA|        |        | 60467|             Algeria|     |        3|  [TMAX, TMIN, PRCP]|         1|     1981|      2019|
|AGM00060468|          |          AG|   35.55|    6.183|   1052.0|              BATNA|        |        | 60468|             Algeria|     |        4|[TMAX, TMIN, PRCP...|         1|     1973|      2020|
|AGM00060507|          |          AG|  35.208|    0.147|    513.9|             GHRISS|        |        | 60507|             Algeria|     |        4|[TMAX, TMIN, PRCP...|         1|     1943|      2020|
|AGM00060511|          |          AG|  35.341|    1.463|    989.1|          BOU CHEKIF|        |        | 60511|             Algeria|     |        4|[TMAX, TMIN, PRCP...|         1|     1983|      2020|
+-----------+----------+------------+--------+---------+---------+--------------------+--------+--------+------+--------------------+-----+---------+--------------------+----------+---------+----------+
only showing top 20 rows
```

Ancy John

Student ID: 52770710

The output is saved in Parquet file format which is more efficient in terms of storage and performance, for Hadoop.

(f) LEFT JOIN your 1000 rows subset of daily and your output from part (e). Are there any stations in your subset of daily that are not in stations at all?

How expensive do you think it would be to LEFT JOIN all of daily and stations? Could you determine if there are any stations in daily that are not in stations without using LEFT JOIN?

Left join daily subset with the stations, returns the below dataframe.

```
In [8]: daily.join(df, on=['ID'], how='left').show()
```

ID	DATE	ELEMENT	VALUE	MEASUREMENT_FLAG	QUALITY_FLAG	SOURCE_FLAG	OBSERVATION_TIME	STATE_CODE	COUNTRY_CODE	LATITUDE	LONGITUDE	ELEVATION	NAME	GSN_FLAG	CRN_FLAG	WMO_ID	COUNTRY	STATE	CORE_TYPE	CORE_ELEMENTS	OTHER_TYPE	FIRSTYEAR	LASTSTYEAR
US1FLSL0019	20200101	PRCP	0.0	null	null	N	null	FL	US	27.3237	-80.3111	4.3	PORT ST. LUCIE 4....				United States	FLORIDA	2	[PRCP, SNOW]	2	2008	2020
US1FLSL0019	20200101	SNOW	0.0	null	null	N	null	FL	US	27.3237	-80.3111	4.3	PORT ST. LUCIE 4....				United States	FLORIDA	2	[PRCP, SNOW]	2	2008	2020
US1NVNY0012	20200101	PRCP	0.0	null	null	N	null	NV	US	36.2127	-116.0355	785.8	PAHRUMP 2.9 SSW				United States	NEVADA	3	[PRCP, SNWD, SNOW]	4	2018	2020
US1NVNY0012	20200101	SNOW	0.0	null	null	N	null	NV	US	36.2127	-116.0355	785.8	PAHRUMP 2.9 SSW				United States	NEVADA	3	[PRCP, SNWD, SNOW]	4	2018	2020
US1ILWM0012	20200101	PRCP	0.0	null	null	N	null	IL	US	37.6283	-89.0039	169.5	MARION 8.0 SSW				United States	ILLINOIS	2	[PRCP, SNOW]	2	2017	2020
USS0018D08S	20200101	TMAX	46.0	null	null	T	null	OR	US	45.19	-118.55	1472.2	County Line				United States	OREGON	4	[TMAX, TMIN, PRCP...	3	1980	2020
USS0018D08S	20200101	TMIN	6.0	null	null	T	null	OR	US	45.19	-118.55	1472.2	County Line				United States	OREGON	4	[TMAX, TMIN, PRCP...	3	1980	2020
USS0018D08S	20200101	TOBS	6.0	null	null	T	null	OR	US	45.19	-118.55	1472.2	County Line				United States	OREGON	4	[TMAX, TMIN, PRCP...	3	1980	2020
USS0018D08S	20200101	PRCP	76.0	null	null	T	null	OR	US	45.19	-118.55	1472.2	County Line				United States	OREGON	4	[TMAX, TMIN, PRCP...	3	1980	2020
USS0018D08S	20200101	SNWD	0.0	null	null	T	null	OR	US	45.19	-118.55	1472.2	County Line				United States	OREGON	4	[TMAX, TMIN, PRCP...	3	1980	2020
USS0018D08S	20200101	TAVG	23.0	null	null	T	null	OR	US	45.19	-118.55	1472.2	County Line				United States	OREGON	4	[TMAX, TMIN, PRCP...	3	1980	2020
USS0018D08S	20200101	WESD	127.0	null	null	T	null	OR	US	45.19	-118.55	1472.2	County Line				United States	OREGON	4	[TMAX, TMIN, PRCP...	3	1980	2020
USC00141761	20200101	TMAX	39.0	null	null	7	0700	KS	US	39.5592	-97.6697	451.4	CONCORDIA 1 W				United States	KANSAS	5	[TMAX, TMIN, PRCP...	10	2003	2020
USC00141761	20200101	TMIN	-50.0	null	null	7	0700	KS	US	39.5592	-97.6697	451.4	CONCORDIA 1 W				United States	KANSAS	5	[TMAX, TMIN, PRCP...	10	2003	2020
USC00141761	20200101	TOBS	-17.0	null	null	7	0700	KS	US	39.5592	-97.6697	451.4	CONCORDIA 1 W				United States	KANSAS	5	[TMAX, TMIN, PRCP...	10	2003	2020
USC00141761	20200101	PRCP	0.0	null	null	7	0700	KS	US	39.5592	-97.6697	451.4	CONCORDIA 1 W				United States	KANSAS	5	[TMAX, TMIN, PRCP...	10	2003	2020
USC00141761	20200101	SNOW	0.0	null	null	7	null	KS	US	39.5592	-97.6697	451.4	CONCORDIA 1 W				United States	KANSAS	5	[TMAX, TMIN, PRCP...	10	2003	2020
USC00141761	20200101	SNWD	0.0	null	null	7	0700	KS	US	39.5592	-97.6697	451.4	CONCORDIA 1 W				United States	KANSAS	5	[TMAX, TMIN, PRCP...	10	2003	2020
USC00141761	20200101	WESD	0.0	null	null	7	null	KS	US	39.5592	-97.6697	451.4	CONCORDIA 1 W				United States	KANSAS	5	[TMAX, TMIN, PRCP...	10	2003	2020
RQC00660061	20200101	TMAX	289.0	null	null	7	0800	PR	RQ	18.1747	-66.7978	557.8	ADJUNTAS SUBSTN				Puerto Rico [Unit...	PUERTO RICO	5	[TMAX, TMIN, PRCP...	11	1970	2020

only showing top 20 rows

There are no null values in the stationd columns. So it can be assumed that there are no additional station records in the daily. Left Anti Join is used to find any stations in daily that is not in the enriched stationd dataframe, which returned an empty output.

```
In [7]: daily.join(df, on=['ID'], how='left_anti').show()
+---+----+-------+-----+----------------+------------+-----------+----------------+
| ID|DATE|ELEMENT|VALUE|MEASUREMENT_FLAG|QUALITY_FLAG|SOURCE_FLAG|OBSERVATION_TIME|
+---+----+-------+-----+----------------+------------+-----------+----------------+
+---+----+-------+-----+----------------+------------+-----------+----------------+
```

There are only 318 unique stations in the daily subset, that is briefing repeatedly in all these 1000 rows. We do not need all these informations on a daily base. And, also to minimise the complexity it is better to keep it separately.

Analysis

Q1. First it will be helpful to know more about the stations in our database, before we study the actual daily climate summaries in more detail.

(a) How many stations are there in total? How many stations have been active in 2020?

Ancy John

Student ID: 52770710

How many stations are in each of the GCOS Surface Network (GSN), the US Historical Climatology Network (HCN), and the US Climate Reference Network (CRN)? Are there any stations that are in more than one of these networks?

Altogether there are 115081 unique station IDs in our database. Of these, 28974 have been active in 2020. GCOS Surface Network (GSN) includes 991 stations. 1218 stations in the US Historical Climatology Network(HCN) and 233 in US Climate Reference Network (CRN). 14 stations are in more than one network.

```
In [3]: df.count()
Out[3]: 115081

In [4]: df.select(F.countDistinct("ID")).show()
+------------------+
|count(DISTINCT ID)|
+------------------+
|            115081|
+------------------+
```

```
In [7]: df.filter("LASTYEAR == 2020").count()
Out[7]: 28974
```

```
In [11]: df.filter("GSN_FLAG == 'GSN'").count()
Out[11]: 991

In [12]: df.filter("CRN_FLAG == 'HCN'").count()
Out[12]: 1218

In [13]: df.filter("CRN_FLAG == 'CRN'").count()
Out[13]: 233
```

```
In [20]: df.filter((F.col("CRN_FLAG") == 'CRN') & (F.col("GSN_FLAG") == 'GSN')).count()
Out[20]: 0

In [21]: df.filter((F.col("CRN_FLAG") == 'HCN') & (F.col("GSN_FLAG") == 'GSN')).count()
Out[21]: 14
```

(b) Count the total number of stations in each country, and store the output in countries using the withColumnRenamed command.

Do the same for states and save a copy of each table to your output directory.

Ancy John

Student ID: 52770710

```
In [7]: N_S = (
   ...:     df
   ...:     .select(['COUNTRY', 'ID'])
   ...:     .groupBy("COUNTRY")
   ...:     .agg({'ID': 'count'})
   ...:     .select(
   ...:         F.col('COUNTRY').alias('NAME'),
   ...:         F.col('count(ID)').alias('NO_STATIONS')
   ...:     )
   ... )

In [8]: countries_ps = countries_ps.join(N_S, on=['NAME'], how='left')

In [9]: countries_ps.show()
+--------------------+----+-----------+
|                NAME|CODE|NO_STATIONS|
+--------------------+----+-----------+
| Antigua and Barbuda|  AC|          2|
|United Arab Emirates|  AE|          4|
|         Afghanistan|  AF|          4|
|             Algeria|  AG|         87|
|          Azerbaijan|  AJ|         66|
|             Albania|  AL|          3|
|             Armenia|  AM|         53|
|              Angola|  AO|          6|
|American Samoa [U...|  AQ|         20|
|           Argentina|  AR|        101|
|           Australia|  AS|      17088|
|             Austria|  AU|         13|
|          Antarctica|  AY|        102|
|             Bahrain|  BA|          1|
|            Barbados|  BB|          1|
|            Botswana|  BC|         21|
|Bermuda [United K...|  BD|          2|
|             Belgium|  BE|          1|
|        Bahamas, The|  BF|         39|
|          Bangladesh|  BG|         10|
+--------------------+----+-----------+
only showing top 20 rows
```

```
In [12]: N_S1 = (
    ...:     df
    ...:     .select(['STATE', 'ID'])
    ...:     .groupBy("STATE")
    ...:     .agg({'ID': 'count'})
    ...:     .select(
    ...:         F.col('STATE').alias('NAME'),
    ...:         F.col('count(ID)').alias('NO_STATIONS')
    ...:     )
    ... )

In [13]: states_ps = states_ps.join(N_S1, on=['NAME'], how='left')

In [14]: states_ps.show()
+--------------------+-----+-----------+
|                NAME|CODE|NO_STATIONS|
+--------------------+-----+-----------+
|             ALBERTA|  AB|       1420|
|              ALASKA|  AK|        986|
|             ALABAMA|  AL|        979|
|            ARKANSAS|  AR|        864|
|      AMERICAN SAMOA|  AS|         20|
|             ARIZONA|  AZ|       1453|
|    BRITISH COLUMBIA|  BC|       1683|
|          CALIFORNIA|  CA|       2798|
|            COLORADO|  CO|       4176|
|         CONNECTICUT|  CT|        310|
|DISTRICT OF COLUMBIA|  DC|         14|
|            DELAWARE|  DE|        114|
|             FLORIDA|  FL|       1744|
|          MICRONESIA|  FM|         38|
|             GEORGIA|  GA|       1210|
|                GUAM|  GU|         20|
|              HAWAII|  HI|        733|
|                IOWA|  IA|        864|
|               IDAHO|  ID|        774|
|            ILLINOIS|  IL|       1783|
+--------------------+-----+-----------+
only showing top 20 rows
```

(c) How many stations are there in the Northern Hemisphere only?

Some of the countries in the database are territories of the United States as indicated by the name of the country. How many stations are there in total in the territories of the United States around the world?

There are 89745 stations in the Northern Hemisphere, and 314 stations in the 8 territories of United States.

Ancy John

Student ID: 52770710

```
In [26]: df.filter("LATITUDE > 0").count()
Out[26]: 89745

In [27]: df.filter("COUNTRY_CODE == 'US'").count()
Out[27]: 61867
```

```
In [42]: df=df.where(F.col("COUNTRY").like("%[United States]"))

In [43]: df.show(3)
+-----------+----------+------------+--------+---------+---------+-----------------+--------+--------+------+--------------------+--------------+-------+
|         ID|STATE_CODE|COUNTRY_CODE|LATITUDE|LONGITUDE|ELEVATION|             NAME|GSN_FLAG|CRN_FLAG|WMO_ID|             COUNTRY|         STATE|CORE_TY|
+-----------+----------+------------+--------+---------+---------+-----------------+--------+--------+------+--------------------+--------------+-------+
|AQC00914145|        AS|          AQ|-14.2833|-170.7167|     14.9|    FAGASA TUTUILA|        |        |      |American Samoa [U...|AMERICAN SAMOA|       |
|AQC00914869|        AS|          AQ|-14.3333|-170.7167|      3.0|TAFUNA AP TUTUILA|        |        |      |American Samoa [U...|AMERICAN SAMOA|       |
|AQC00914902|        AS|          AQ|-14.2728|-170.6922|     80.8|          VAIPITO|        |        |      |American Samoa [U...|AMERICAN SAMOA|       |
+-----------+----------+------------+--------+---------+---------+-----------------+--------+--------+------+--------------------+--------------+-------+
only showing top 3 rows


In [44]: df.select("ID").distinct().count()
Out[44]: 314

In [45]: df.select("COUNTRY").distinct().count()
Out[45]: 8
```

Q2. You can create user defined functions in Spark by taking native Python functions and wrapping them using pyspark.sql.functions.udf (which can take multiple columns as inputs). You will find this functionality extremely useful.

(a) Write a Spark function that computes the geographical distance between two stations using their latitude and longitude as arguments. You can test this function by using CROSSJOIN on a small subset of stations to generate a table with two stations in each row.

Note that there is more than one way to compute geographical distance, choose a method that at least takes into account that the earth is spherical.

Geometrical distance between the two stations is calculated using Haversine formula.

$$d = 2rsin^{-1}\left(\sqrt{sin^2\left(\frac{\Phi_2-\Phi_1}{2}\right) + cos(\Phi_1)cos(\Phi_2)sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)}\right)$$

Python function haversine() is converted to a user defined function to operate on a spark dataframe. This udf is applied to the cross joined dataframe to get the desired output.

```
In [76]: GD_test.show()
+-----------+--------------+------+------+-----------+--------------+------+------+--------+
|        ID1|         NAME1|  LAT1|  LON1|        ID2|         NAME2|  LAT2|  LON2|DISTANCE|
+-----------+--------------+------+------+-----------+--------------+------+------+--------+
|IN003020100|        TEZPUR|26.617|92.783|IN003050401|NORTH LAKHIMPUR|27.233|94.117|  148.94|
|IN003020100|        TEZPUR|26.617|92.783|IN004051800|          GAYA| 24.75| 84.95|  811.76|
|IN003020100|        TEZPUR|26.617|92.783|IN005030100|         DEESA|  24.2|  72.2| 2082.43|
|IN003020100|        TEZPUR|26.617|92.783|IN005100400|     PORBANDAR| 21.65|69.667| 2406.09|
|IN003050401|NORTH LAKHIMPUR|27.233|94.117|IN003020100|        TEZPUR|26.617|92.783|  148.94|
|IN003050401|NORTH LAKHIMPUR|27.233|94.117|IN004051800|          GAYA| 24.75| 84.95|  956.63|
|IN003050401|NORTH LAKHIMPUR|27.233|94.117|IN005030100|         DEESA|  24.2|  72.2| 2218.47|
|IN003050401|NORTH LAKHIMPUR|27.233|94.117|IN005100400|     PORBANDAR| 21.65|69.667| 2547.03|
|IN004051800|          GAYA| 24.75| 84.95|IN003020100|        TEZPUR|26.617|92.783|  811.76|
|IN004051800|          GAYA| 24.75| 84.95|IN003050401|NORTH LAKHIMPUR|27.233|94.117|  956.63|
|IN004051800|          GAYA| 24.75| 84.95|IN005030100|         DEESA|  24.2|  72.2| 1291.32|
|IN004051800|          GAYA| 24.75| 84.95|IN005100400|     PORBANDAR| 21.65|69.667| 1598.56|
|IN005030100|         DEESA|  24.2|  72.2|IN003020100|        TEZPUR|26.617|92.783| 2082.43|
|IN005030100|         DEESA|  24.2|  72.2|IN003050401|NORTH LAKHIMPUR|27.233|94.117| 2218.47|
|IN005030100|         DEESA|  24.2|  72.2|IN004051800|          GAYA| 24.75| 84.95| 1291.32|
|IN005030100|         DEESA|  24.2|  72.2|IN005100400|     PORBANDAR| 21.65|69.667|  384.28|
|IN005100400|     PORBANDAR| 21.65|69.667|IN003020100|        TEZPUR|26.617|92.783| 2406.09|
|IN005100400|     PORBANDAR| 21.65|69.667|IN003050401|NORTH LAKHIMPUR|27.233|94.117| 2547.03|
|IN005100400|     PORBANDAR| 21.65|69.667|IN004051800|          GAYA| 24.75| 84.95| 1598.56|
|IN005100400|     PORBANDAR| 21.65|69.667|IN005030100|         DEESA|  24.2|  72.2|  384.28|
+-----------+--------------+------+------+-----------+--------------+------+------+--------+
```

(b) Apply this function to compute the pairwise distances between all stations in NewZealand, and save the result to your output directory.

Ancy John

Student ID: 52770710

What two stations are the geographically furthest apart in New Zealand?

```
In [92]: GD_NZ = GD_NZ.orderBy('DISTANCE', ascending = False)

In [93]: GD_NZ.show()
+-----------+------------------+-------+--------+-----------+------------------+-------+--------+--------+
|        ID1|             NAME1|   LAT1|    LON1|        ID2|             NAME2|   LAT2|    LON2|DISTANCE|
+-----------+------------------+-------+--------+-----------+------------------+-------+--------+--------+
|NZ000939450|CAMPBELL ISLAND AWS| -52.55| 169.167|NZ000093994| RAOUL ISL/KERMADEC| -29.25|-177.917| 2799.18|
|NZM00093929| ENDERBY ISLAND AWS|-50.483|   166.3|NZ000093994| RAOUL ISL/KERMADEC| -29.25|-177.917| 2705.42|
|NZ000093844|INVERCARGILL AIRPOR|-46.417| 168.333|NZ000093994| RAOUL ISL/KERMADEC| -29.25|-177.917| 2251.34|
|NZ000093994| RAOUL ISL/KERMADEC| -29.25|-177.917|NZ000937470|         TARA HILLS|-44.517|   169.9| 2008.89|
|NZ000093012|           KAITAIA|  -35.1| 173.267|NZ000939450|CAMPBELL ISLAND AWS| -52.55| 169.167| 1967.22|
|NZ000093012|           KAITAIA|  -35.1| 173.267|NZM00093929| ENDERBY ISLAND AWS|-50.483|   166.3| 1800.52|
|NZ000093994| RAOUL ISL/KERMADEC| -29.25|-177.917|NZM00093781|  CHRISTCHURCH INTL|-43.489| 172.532| 1796.56|
|NZ000936150| HOKITIKA AERODROME|-42.717| 170.983|NZ000093994| RAOUL ISL/KERMADEC| -29.25|-177.917| 1796.36|
|NZM00093110|  AUCKLAND AERO AWS|  -37.0|   174.8|NZ000939450|CAMPBELL ISLAND AWS| -52.55| 169.167| 1783.96|
|NZ000939450|CAMPBELL ISLAND AWS| -52.55| 169.167|NZ000093292| GISBORNE AERODROME| -38.65| 177.983| 1687.99|
|NZM00093678|         KAIKOURA|-42.417|   173.7|NZ000093994| RAOUL ISL/KERMADEC| -29.25|-177.917| 1645.56|
|NZM00093110|  AUCKLAND AERO AWS|  -37.0|   174.8|NZM00093929| ENDERBY ISLAND AWS|-50.483|   166.3| 1644.84|
|NZ000093994| RAOUL ISL/KERMADEC| -29.25|-177.917|NZ000939870|CHATHAM ISLANDS AWS| -43.95|-176.567| 1638.94|
|NZM00093929| ENDERBY ISLAND AWS|-50.483|   166.3|NZ000093292| GISBORNE AERODROME| -38.65| 177.983|  1604.5|
|NZ000939450|CAMPBELL ISLAND AWS| -52.55| 169.167|NZ000933090|   NEW PLYMOUTH AWS|-39.017| 174.183| 1553.29|
|NZ000093994| RAOUL ISL/KERMADEC| -29.25|-177.917|NZM00093439|WELLINGTON AERO AWS|-41.333|   174.8| 1495.94|
|NZM00093929| ENDERBY ISLAND AWS|-50.483|   166.3|NZ000939870|CHATHAM ISLANDS AWS| -43.95|-176.567| 1478.94|
|NZ000093994| RAOUL ISL/KERMADEC| -29.25|-177.917|NZ000093417|    PARAPARAUMU AWS|  -40.9| 174.983| 1446.32|
|NZ000939450|CAMPBELL ISLAND AWS| -52.55| 169.167|NZ000939870|CHATHAM ISLANDS AWS| -43.95|-176.567| 1420.22|
|NZM00093929| ENDERBY ISLAND AWS|-50.483|   166.3|NZ000933090|   NEW PLYMOUTH AWS|-39.017| 174.183| 1416.91|
+-----------+------------------+-------+--------+-----------+------------------+-------+--------+--------+
only showing top 20 rows
```

Duplicate rows are removed using the dropDuplicates() method. The output data frame holds 105 records. Geographically furthest stations in New Zealand is `CAMPBELL ISLAND AWS` and `RAOUL ISL/KERMADEC`.

Q3. Next you will start exploring all of the daily climate summaries in more detail. In order to know how efficiently you can load and apply transformations to daily, you need to first understand the level of parallelism that you can achieve for the specific structure and format of daily.

(a). Recall the hdfs commands that you used to explore the data in Processing Q1. You would have used

hdfs dfs -ls [path] hdfs dfs -du [path]

to determine the size of files under a specific path in HDFS.

Use the following command

hdfs getconf -confKey "dfs.blocksize"

to determine the default blocksize of HDFS. How many blocks are required for the daily climate summaries for the year 2020? What about the year 2010? What are the individual block sizes for the year 2010? You will need to use another hdfs command.

Based on these results, is it possible for Spark to load and apply transformations in parallel for the year 2020? What about the year 2010?

Files in HDFS are broken into data blocks and stored as individual units. The default size of a data block is 128MB. HDFS data blocks are efficient in handling file sizes, providing fault tolerance and high

availability. Moreover, this is a simple concept and the metadata files are stored separately. Parallelism is achieved using the worker nodes.

```
[ajo139@canterbury.ac.nz@mathmadslinux1p ~]$ hdfs dfs -du /data/ghcnd/daily/2020.csv.gz
31626590  253012720  /data/ghcnd/daily/2020.csv.gz
[ajo139@canterbury.ac.nz@mathmadslinux1p ~]$ hdfs dfs -du /data/ghcnd/daily/2010.csv.gz
232080599  1856644792  /data/ghcnd/daily/2010.csv.gz
[ajo139@canterbury.ac.nz@mathmadslinux1p ~]$ hdfs getconf -confKey "dfs.blocksize"
134217728
```

Daily climate summaries for the year 2020 can be held in one single block. According to the default block size we have for now, 2 blocks were required for the year 2010.

```
[ajo139@canterbury.ac.nz@mathmadslinux1p ~]$  hdfs fsck /data/ghcnd/daily/2010.csv.gz -files -blocks
Connecting to namenode via http://node0:9870/fsck?ugi=ajo139&files=1&blocks=1&path=%2Fdata%2Fghcnd%2Fdaily%2F2010.csv.gz
FSCK started by ajo139 (auth:SIMPLE) from /192.168.40.10 for path /data/ghcnd/daily/2010.csv.gz at Sat Apr 18 01:06:00 NZST 2020
/data/ghcnd/daily/2010.csv.gz 232080599 bytes, replicated: replication=8, 2 block(s):  OK
0. BP-1663138130-132.181.39.102-1551363950352:blk_1073822197_81375 len=134217728 Live_repl=8
1. BP-1663138130-132.181.39.102-1551363950352:blk_1073822198_81376 len=97862871 Live_repl=8
```

The 2010 file is broken down into 2 data blocks and had the same default block size, 134217728. Both files are located in the same system with same number of data nodes. Spark uses Resilient Distributed Datasets (RDD) to perform parallel processing across these nodes. Irrespective of the number of data blocks Spark can load the data and apply transformations on it, with the same efficiency.

(b) Load and count the number of observations in daily for each of the years 2015 and 2020.

How many tasks were executed by each stage of each job?

You can check this by using your application console. which you can find in the web user interface (mathmadslinux1p:8080). You can either determine the number of tasks executed by each stage of each job or determine the total number of tasks completed by each executor after the job has completed.

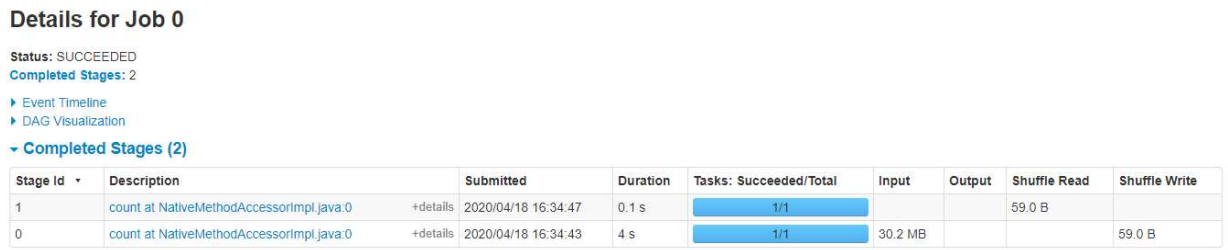
Did the number of tasks executed correspond to the number of blocks in each input?

```
   ...: daily2020.count()
Out[3]: 5215365

In [4]: daily2015 = (
   ...:     spark.read.format("com.databricks.spark.csv")
   ...:     .option("header", "false")
   ...:     .option("inferSchema", "false")
   ...:     .schema(schema_daily)
   ...:     .load("hdfs:///data/ghcnd/daily/2015.csv.gz")
   ...: )
   ...: daily2015.count()
Out[4]: 34899014
```

Ancy John

Student ID: 52770710



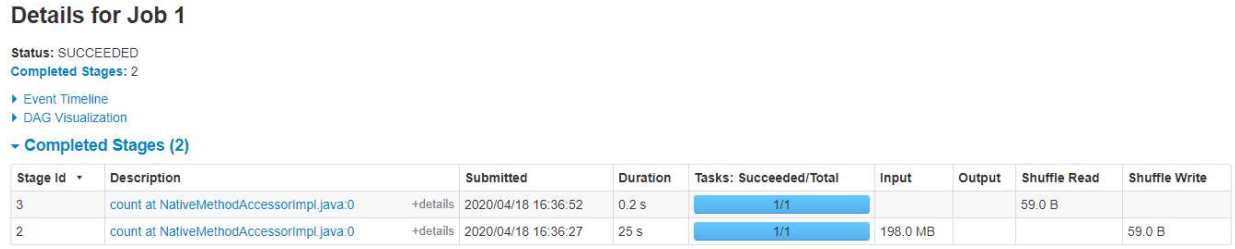
Details for Job 0

Status: SUCCEEDED
Completed Stages: 2

▸ Event Timeline
▸ DAG Visualization

▾ Completed Stages (2)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
1	count at NativeMethodAccessorImpl.java:0 +details	2020/04/18 16:34:47	0.1 s	1/1			59.0 B	
0	count at NativeMethodAccessorImpl.java:0 +details	2020/04/18 16:34:43	4 s	1/1	30.2 MB			59.0 B



Details for Job 1

Status: SUCCEEDED
Completed Stages: 2

▸ Event Timeline
▸ DAG Visualization

▾ Completed Stages (2)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
3	count at NativeMethodAccessorImpl.java:0 +details	2020/04/18 16:36:52	0.2 s	1/1			59.0 B	
2	count at NativeMethodAccessorImpl.java:0 +details	2020/04/18 16:36:27	25 s	1/1	198.0 MB			59.0 B

There were 2 stages for each of the 2 jobs. The first stage executed writing in 1 task and the second stage executed reading in 1 another task tasks.

The storage and the processing are the 2 separate core components on Hadoop. Thus the number of tasks executed is not corresponding to the number of input blocks.

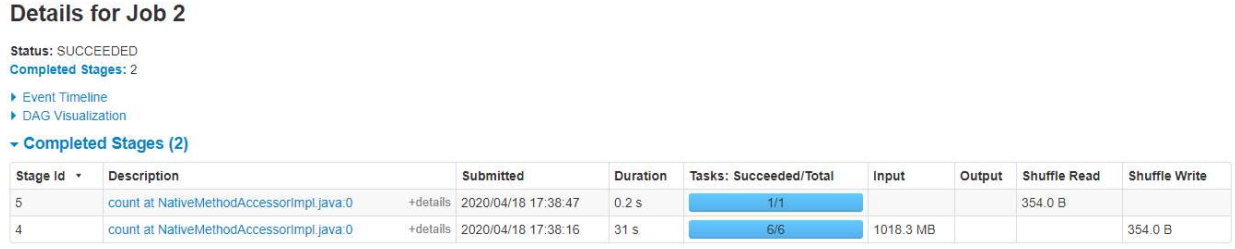
(c) Load and count the number of observations in daily from 2015 to 2020.

Note that you can use any regular expressions in the path argument of the read command.

Now how many tasks were executed by each stage, and how does this number correspond to your input?

Find out how Spark partitions input files that are compressed.

```
In [5]: daily5ys = (
   ...:     spark.read.format("com.databricks.spark.csv")
   ...:     .option("header", "false")
   ...:     .option("inferSchema", "false")
   ...:     .schema(schema_daily)
   ...:     .load("hdfs:///data/ghcnd/daily/{201[5-9],2020}.csv.gz")
   ...: )
   ...: daily5ys.count()
Out[5]: 178918901
```



Details for Job 2

Status: SUCCEEDED
Completed Stages: 2

▸ Event Timeline
▸ DAG Visualization

▾ Completed Stages (2)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
5	count at NativeMethodAccessorImpl.java:0 +details	2020/04/18 17:38:47	0.2 s	1/1			354.0 B	
4	count at NativeMethodAccessorImpl.java:0 +details	2020/04/18 17:38:16	31 s	6/6	1018.3 MB			354.0 B

Ancy John

Student ID: 52770710

The initial stage executed 6 tasks and the second stage executed 1 task for this job. The input size for this job is 1018MB, that demanded more resources and transformation tasks to achieve parallel processing. The job is done with 4 executors together. The output is generated in 31 seconds just 5 seconds more than the previous job that had an input size of just 198MB. More executors on the cluster is important to increase parallelism.

Compressed files reduce the file sizes and are helpful in handling data transfers across the nodes. When it comes to distributed computing partitioning of these files is a challenge, when the format is not splittable. Apache spark uses the codecs provided by Hadoop to deal with these files. Splittable compressed files are processed as any other file. Not splittable compressed files are processed by a single executor. (waitingforcode.com)

(d) Based on parts (b) and (c), what level of parallelism can you achieve when loading and applying transformations to daily? Can you think of any way you could increase this level of parallelism either in Spark or by additional preprocessing?

The daily data set is 15.5 GB. Improved level of parallelism can be obtained by more executors and more resources. Another method is by way of managing spark partitions.

For large volume of data processing in spark, data partitioning is critical. An atomic chunk of data stored on a node in the cluster is called a partition in spark and are basic units of parallelism. Apache Spark manages data through RDDs. By default a partition is created for every HDFS partition of size 64MB. Apache Spark can run a single concurrent task for every partition of an RDD, up to the total number of cores in the cluster. The best way to decide on the number of partitions in an RDD is to make the number of partitions equal to the number of cores in the cluster so that all the partitions will process in parallel and the resources will be utilized in an optimal way. (ProjectPro) We need to understand how data is partitioned and repartition method can be used to modify it.

```
In [6]: print(daily5ys.rdd.getNumPartitions())
6

In [7]: print(daily2015.rdd.getNumPartitions())
1
```

Q4 All of the data stored in HDFS under hdfs:///data/ghcnd has a replication factor of 8 and is available locally on every node in our cluster. As such you will always be able to load and apply transformations to multiple years of daily in parallel.

Again,youmaywanttouseonlypartofthedailyclimatesummarieswhileyouarestilldeveloping your code so that you can use fewer resources to get preliminary results without waiting all day.

(a) Count the number of rows 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 never try to cache or collect all of daily.

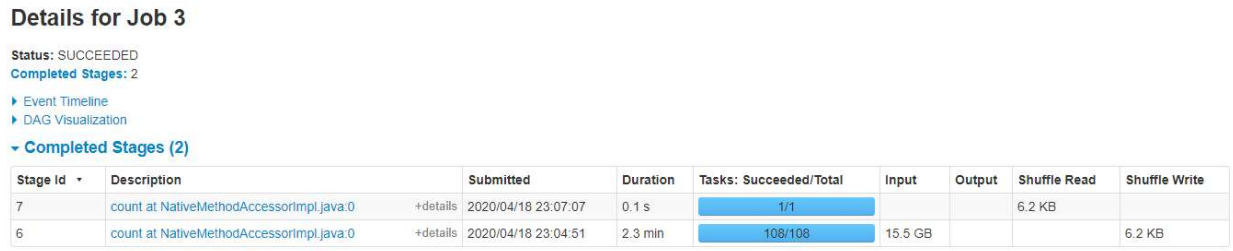
Ancy John

Student ID: 52770710

```
In [8]: daily = (
   ...:     spark.read.format("com.databricks.spark.csv")
   ...:     .option("header", "false")
   ...:     .option("inferSchema", "false")
   ...:     .schema(schema_daily)
   ...:     .load("hdfs:///data/ghcnd/daily")
   ...: )

In [9]: daily.count()
Out[9]: 2928664523

In [10]: print(daily.rdd.getNumPartitions())
108
```



Details for Job 3

Status: SUCCEEDED
Completed Stages: 2

▸ Event Timeline
▸ DAG Visualization

▾ Completed Stages (2)

Stage Id	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
7	count at NativeMethodAccessorImpl.java:0	+details	2020/04/18 23:07:07	0.1 s	1/1			6.2 KB	
6	count at NativeMethodAccessorImpl.java:0	+details	2020/04/18 23:04:51	2.3 min	108/108	15.5 GB			6.2 KB

15.5 GB of data has got 108 partitions. Apparently, number of partitions is same as the number of tasks performed in stage 1. This is perfectly in concordance with the number of cores, where each of the 8 cores have to work on 12 partitions each. The job is executed in 2 stages and 109 tasks in 2.4 minutes. This is the level of parallelism we achieved using 4 executors, 8 cores and 4GB each of the worker and master memories. If we double the resources, the task will be accomplished half the time.

(b) Filter daily using the filter command to obtain the subset of observations containing the five core elements described in inventory.

How many observations are there for each of the five core elements?

Which element has the most observations? Is this result consistent with Processing Q3?

```
In [11]: daily.filter('ELEMENT == "PRCP"').count()
Out[11]: 1021682210

In [12]: daily.filter('ELEMENT == "SNOW"').count()
Out[12]: 332430532

In [13]: daily.filter('ELEMENT == "SNWD"').count()
Out[13]: 283572167

In [14]: daily.filter('ELEMENT == "TMAX"').count()
Out[14]: 436709350

In [15]: daily.filter('ELEMENT == "TMIN"').count()
Out[15]: 435296249
```

Ancy John

Student ID: 52770710

```
In [19]: df = (
    ...:     spark.read.format("parquet")
    ...:     .option("header", "true")
    ...:     .option("inferSchema", "true")
    ...:     .load("hdfs:///user/ajo139/outputs/ghcnd/stations.parquet")
    ...: )
    ...:
    ...: df = df.select(['ID', 'CORE_ELEMENTS']).withColumn('PRCP', array_contains(df.CORE_ELEMENTS, 'PRCP'))

In [20]:  df.filter('PRCP == True').count()
Out[20]: 79009

In [21]: df.show(3)
+-----------+--------------------+----+
|         ID|       CORE_ELEMENTS|PRCP|
+-----------+--------------------+----+
|AE000041196|  [TMAX, TMIN, PRCP]|true|
|AEM00041218|  [TMAX, TMIN, PRCP]|true|
|AFM00040938|[TMAX, TMIN, PRCP...|true|
+-----------+--------------------+----+
only showing top 3 rows
```

```
In [16]: df.show(3)
+-----------+--------------------+------------+------------+
|         ID|       CORE_ELEMENTS|TEMPERATURE1|TEMPERATURE2|
+-----------+--------------------+------------+------------+
|AE000041196|  [TMAX, TMIN, PRCP]|        true|        true|
|AEM00041218|  [TMAX, TMIN, PRCP]|        true|        true|
|AFM00040938|[TMAX, TMIN, PRCP...|        true|        true|
+-----------+--------------------+------------+------------+
only showing top 3 rows


In [17]: df.filter('TEMPERATURE1 == True').count()
Out[17]: 30335

In [18]: df.filter('TEMPERATURE2 == True').count()
Out[18]: 30238
```

The precipitation element PRCP has the most observations. Out of the 115081 unique stations in the data 79009 stations were collecting PRCP from Q3 part of Processing. Only 30335 & 30238 stations were collecting data on TMAX and TMIN respectively.

(c) Many stations collect TMAX and TMIN, but do not necessarily report them simultaneously due to issues with data collection or coverage. Determine how many observations of TMIN do not have a corresponding observation of TMAX.

How many different stations contributed to these observations?

I filtered out the other elements, used the groupBy() function on ID and DATE column to produce the distinct combinations of these columns, and then used the pivot() function to pivote the ELEMENT column with its VALUE. The output is again filtered for the TMAX column with null values where TMIN is not a null value.

Ancy John

Student ID: 52770710

```
In [20]: daily1 = (
    ...:     daily
    ...:     .select(['ID','DATE','ELEMENT','VALUE'])
    ...:     .filter((F.col("ELEMENT") == 'TMAX') | (F.col("ELEMENT") == 'TMIN'))
    ...:     .groupBy('ID','DATE')
    ...:     .pivot('ELEMENT',['TMAX', 'TMIN'])
    ...:     .agg(first("VALUE"))
    ...: )
    ...: daily1.show(5)
    ...: daily1 = daily1.where((F.col("TMAX").isNull()) & (F.col("TMIN").isNotNull()))
    ...: daily1.show(5)
    ...: daily1.count()
    ...:
    ...: from pyspark.sql.functions import countDistinct
    ...: daily1.agg(countDistinct('ID', 'DATE')).show() # To check if the columns are distinct
+-----------+--------+-----+-----+
|         ID|    DATE| TMAX| TMIN|
+-----------+--------+-----+-----+
|ACW00011604|19490121|278.0|217.0|
|ACW00011604|19490314|278.0|206.0|
|ACW00011604|19490316|283.0|222.0|
|ACW00011604|19490401|278.0|194.0|
|ACW00011604|19490409|283.0|233.0|
+-----------+--------+-----+-----+
only showing top 5 rows

+-----------+--------+----+-----+
|         ID|    DATE|TMAX| TMIN|
+-----------+--------+----+-----+
|AE000041196|19440725|null|280.0|
|AE000041196|19450521|null|224.0|
|AE000041196|19550108|null|128.0|
|AE000041196|19570320|null|144.0|
|AE000041196|19581208|null|144.0|
+-----------+--------+----+-----+
only showing top 5 rows
```

```
In [21]: daily1.count()
Out[21]: 8428801

In [22]: daily1.agg(countDistinct('ID')).show()
+------------------+
|count(DISTINCT ID)|
+------------------+
|             27526|
+------------------+
```

8428801 observations of TMIN do not have a corresponding observation of TMAX and 27526 stations contributed to these observations.

(d) Filter daily to obtain all observations of TMIN and TMAX for all stations in New Zealand and save the result to your output directory.

How many observations are there, and how many years are covered by the observations?

Use hdfs dfs -copyToLocal to copy the output from HDFS to your local home directory and count the number of rows in the part files using the wc -l bash command. This should match the number of observations that you counted using Spark.

Plot time series for TMIN and TMAX on the same axis for each station in New Zealand using Python, R, or any other programming language you know well. Also, plot the average time series for TMIN and TMAX for the entire country.

```
In [43]: daily_NZ = (
    ...:     daily
    ...:     .withColumn('COUNTRY_CODE', F.substring(daily['ID'],1,2)) # To extract the country code
    ...:     .filter(((F.col("ELEMENT") == 'TMAX') | (F.col("ELEMENT") == 'TMIN')) & (F.col("COUNTRY_CODE") == 'NZ'))
    ...:     .drop('COUNTRY_CODE')
    ...: )

In [44]: daily_NZ.write.parquet('hdfs:///user/ajo139/outputs/ghcnd/daily_NZ.parquet')

In [45]: daily_NZ.count()
Out[45]: 458892
```

Ancy John

Student ID: 52770710

I have used distinc() on column COUNTRY name to see if there any territories included.

```
In [11]: df.select("COUNTRY").distinct().count()
Out[11]: 1

In [12]: df.select("COUNTRY").distinct().show()
+-----------+
|    COUNTRY|
+-----------+
|New Zealand|
+-----------+
```

```
In [58]: year1 = daily_test.agg({'YEAR': 'max'}).collect()[0]

In [59]: year2 = daily_test.agg({'YEAR': 'min'}).collect()[0]
```

```
In [67]: Range = year1['max(YEAR)'] - year2['min(YEAR)']

In [68]: print(year1['max(YEAR)'])
2020

In [69]: print(year2['min(YEAR)'])
1940

In [70]: print(Range)
80
```

There are 458892 observations in the daily New Zealand file. The observations are covered from the year 1940 to 2020 (80years).

```
[ajo139@canterbury.ac.nz@mathmadslinux1p ~]$ hdfs dfs -cat /user/ajo139/outputs/ghcnd/daily_NZ.csv.gz/*.csv.gz | gunzip | wc -l
458973
[ajo139@canterbury.ac.nz@mathmadslinux1p ~]$ hdfs dfs -copyToLocal /user/ajo139/outputs/ghcnd/daily_NZ.csv.gz /users/home/ajo139/outputs/data
[ajo139@canterbury.ac.nz@mathmadslinux1p ~]$ cat /users/home/ajo139/outputs/data/daily_NZ.csv.gz/*.csv.gz | gunzip | wc -l
458973
```

I could not count the rows from my parquet file, that gave me count of just 1-part file (4362) even after trying all the ways I could. I tried minimizing the number of partitions using coalesce(1) function. I even tried installing the parquet tools but failed. Finally, I changed the format to .csc.gz.

However, when I checked the number of rows using wc -l bash command, the result is slightly different 458973. This add on is due to the extra headers in each part file.

To plot the time series, I have taken the yearly average of values, and joined with the stations to get the station names.

Ancy John

Student ID: 52770710

```
In [21]: df_st = (
    ...:       spark.read.format("parquet")
    ...:       .option("header", "true")
    ...:       .option("inferSchema", "true")
    ...:       .load("hdfs:///user/ajo139/outputs/ghcnd/stations.parquet")
    ...:  )
    ...: df_st = df_st.select(['ID', 'NAME', 'COUNTRY'])
    ...: df = df.join(df_st, on=['ID'], how='left')

In [22]: df.show(5)
+-----------+----+-------+-----------------+-----------------+-----------+
|         ID|YEAR|ELEMENT|    AVERAGE_VALUE|             NAME|    COUNTRY|
+-----------+----+-------+-----------------+-----------------+-----------+
|NZ000093292|1978|   TMIN| 88.12054794520547|GISBORNE AERODROME|New Zealand|
|NZ000937470|1978|   TMIN|41.276712328767125|        TARA HILLS|New Zealand|
|NZ000093417|1978|   TMIN| 92.81593406593407|   PARAPARAUMU AWS|New Zealand|
|NZ000933090|1978|   TMAX| 178.8904109589041|  NEW PLYMOUTH AWS|New Zealand|
|NZ000936150|1973|   TMIN| 72.88493150684931|HOKITIKA AERODROME|New Zealand|
+-----------+----+-------+-----------------+-----------------+-----------+
only showing top 5 rows
```

```
In [13]: df.select("NAME").distinct().show()
+------------------+
|              NAME|
+------------------+
| GISBORNE AERODROME|
| RAOUL ISL/KERMADEC|
|         TARA HILLS|
|CAMPBELL ISLAND AWS|
|  AUCKLAND AERO AWS|
| ENDERBY ISLAND AWS|
|  CHRISTCHURCH INTL|
|WELLINGTON AERO AWS|
|INVERCARGILL AIRPOR|
|    PARAPARAUMU AWS|
|   NEW PLYMOUTH AWS|
| HOKITIKA AERODROME|
|            KAITAIA|
|CHATHAM ISLANDS AWS|
|           KAIKOURA|
+------------------+
```

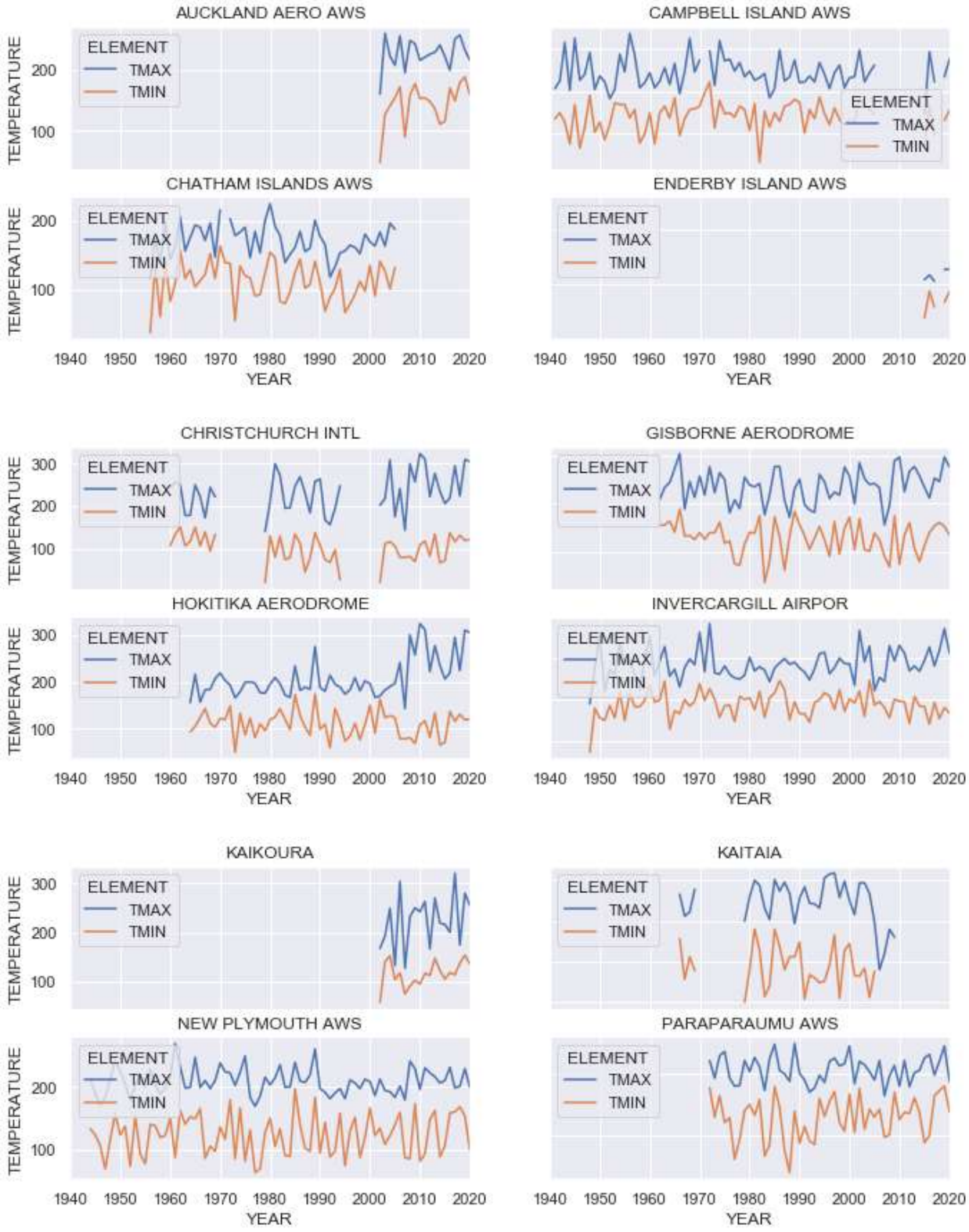
Time series temperature for each station in New Zealand. I used pandas and matplotlib from python, for data visualization.

```
fig, axs = plt.subplots(2,2)
df.groupby(['YEAR','ELEMENT']).mean()['AUCKLAND AERO AWS'].unstack().plot(ax = axs[0, 0])
axs[0, 0].set_title('AUCKLAND AERO AWS')
df.groupby(['YEAR','ELEMENT']).mean()['CAMPBELL ISLAND AWS'].unstack().plot(ax = axs[0, 1])
axs[0, 1].set_title('CAMPBELL ISLAND AWS')
df.groupby(['YEAR','ELEMENT']).mean()['CHATHAM ISLANDS AWS'].unstack().plot(ax = axs[1, 0])
axs[1, 0].set_title('CHATHAM ISLANDS AWS')
df.groupby(['YEAR','ELEMENT']).mean()['ENDERBY ISLAND AWS'].unstack().plot(ax = axs[1, 1])
axs[1, 1].set_title('ENDERBY ISLAND AWS')
for ax in axs.flat:
    ax.set(xlabel='YEAR', ylabel='TEMPERATURE')

# Hide x labels and tick labels for top plots and y ticks for right plots.
for ax in axs.flat:
    ax.label_outer()
```

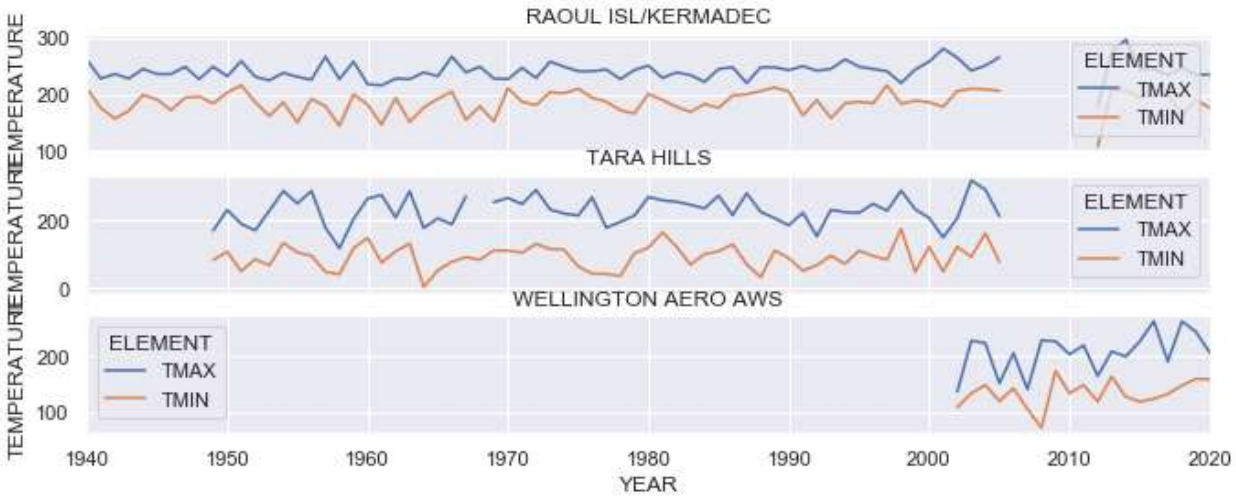
Ancy John

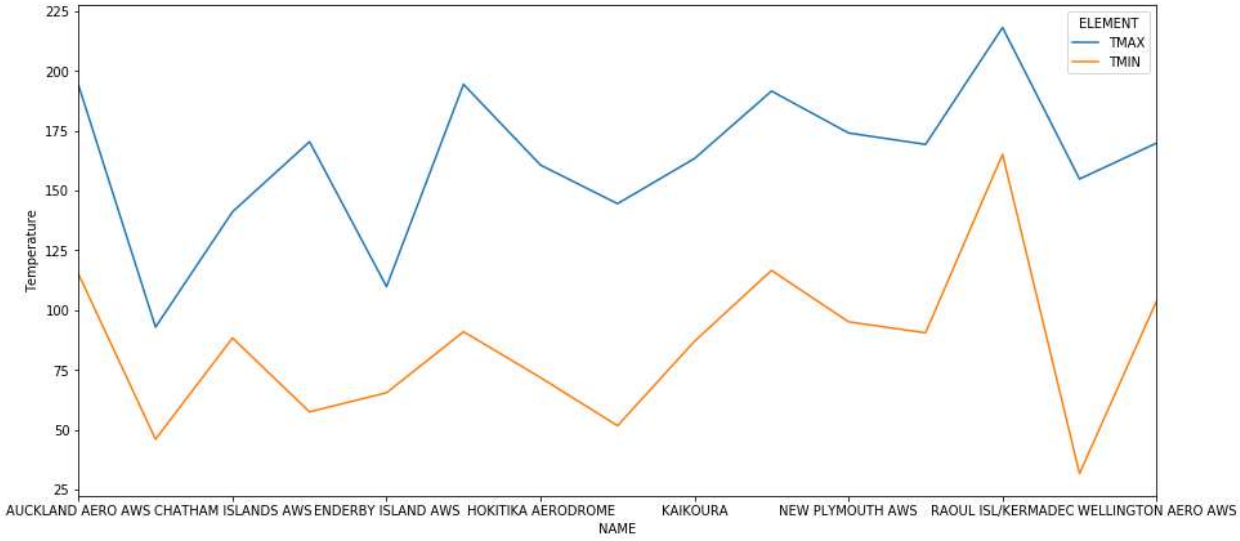
Student ID: 52770710



Ancy John

Student ID: 52770710



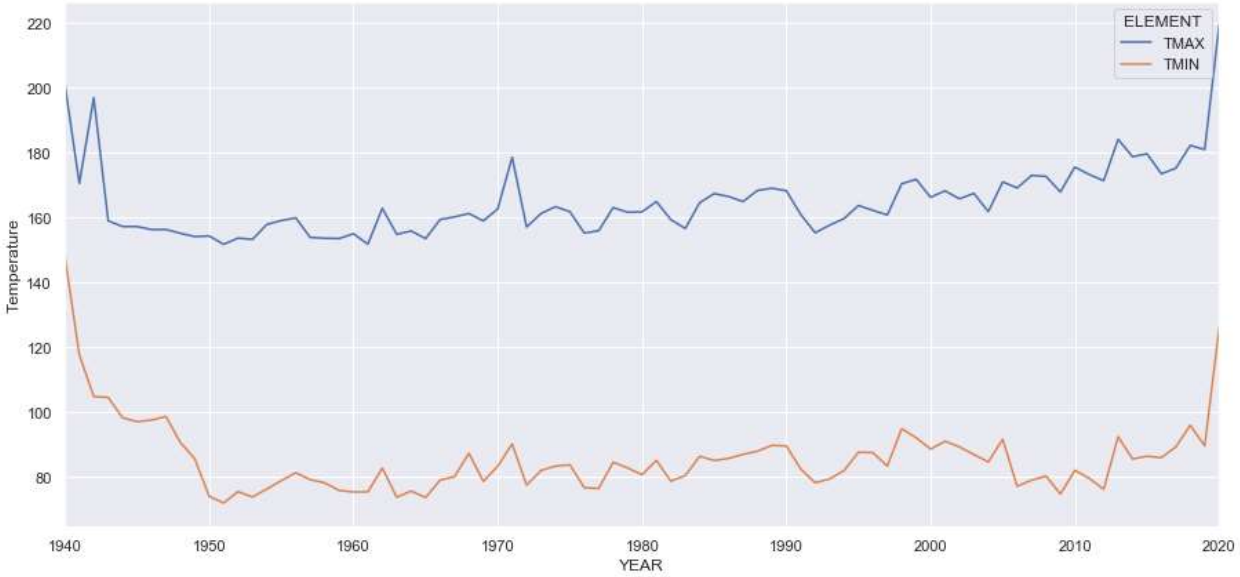


Time series temperature for the entire New Zealand

```
# plot
#df1 = df1.set_index('YEAR')
fig, ax = plt.subplots(figsize=(15,7))
ax.set_ylabel('Temperature');
df1.groupby(['YEAR','ELEMENT']).mean()['VALUE'].unstack().plot(ax=ax)
```

Ancy John

Student ID: 52770710



(e) Group the precipitation observations by year and country. Compute the average rainfall in each year for each country and save this result to your output directory.

Which country has the highest average rainfall in a single year across the entire dataset? Is this result sensible? Is this result consistent?

Find an elegant way to plot the cumulative rainfall for each country using Python, R, or any other programming language you know well. There are many ways to do this in Python and R specifically, such as using a choropleth to color a map according to average rainfall.

Average rainfall groupby year and countrycode is obtained from daily. This file is joined with the station data to obtain the country name.

```
In [100]: from pyspark.sql.functions import avg
     ...: daily_prcp = (
     ...:     daily
     ...:     .where('ELEMENT == "PRCP"')
     ...:     .withColumn('COUNTRY_CODE', F.substring(daily['ID'],1,2))
     ...:     .withColumn('YEAR', F.substring(daily['DATE'],1,4))
     ...:     .groupBy('COUNTRY_CODE', 'YEAR')
     ...:     .agg(avg('VALUE').alias('AVERAGE_PRCP'))
     ...: )
     ...: daily_prcp.write.parquet('hdfs:///user/ajo139/outputs/ghcnd/daily_prcp.parquet')
```

```
In [14]: df.show(1)
+----+----+------------------+------+
|CODE|YEAR|      AVERAGE_PRCP|  NAME|
+----+----+------------------+------+
|  SW|2008|19.859994571703705|Sweden|
+----+----+------------------+------+
only showing top 1 row
```

Ancy John

Student ID: 52770710

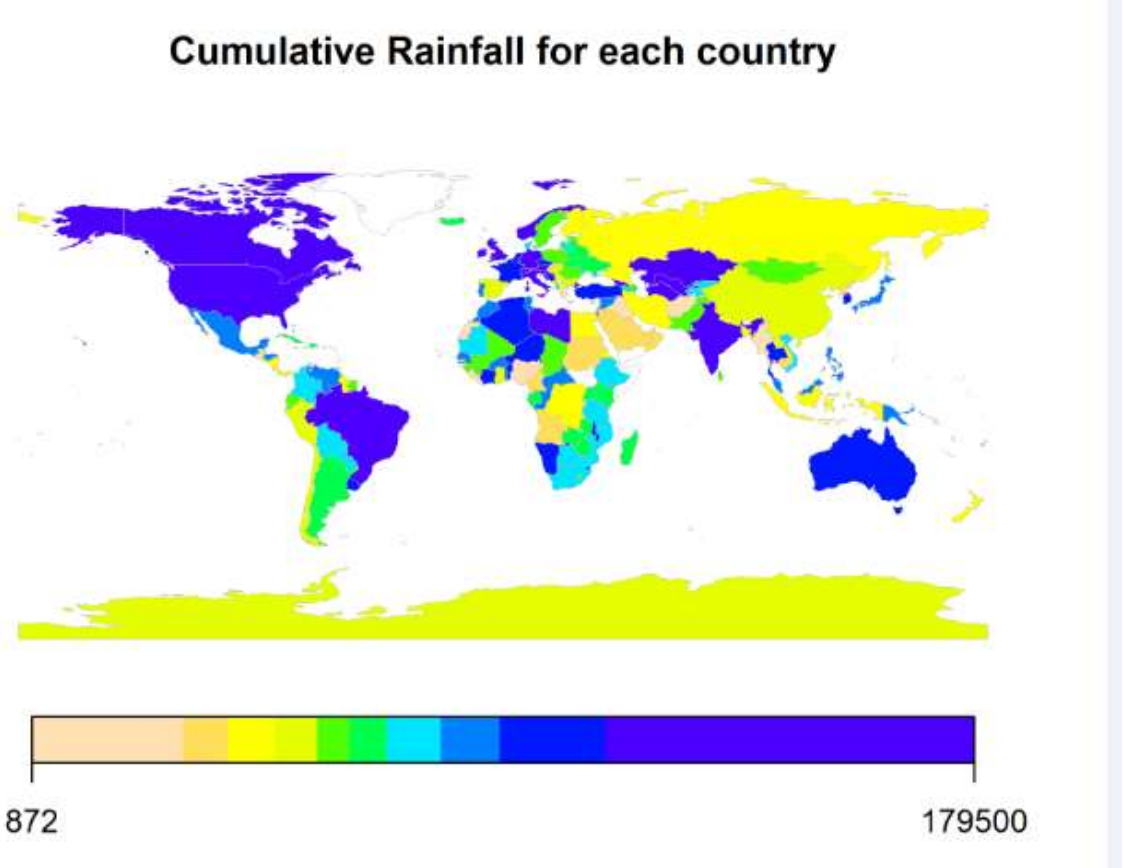
```
In [111]: top = daily_prcp.agg({'AVERAGE_PRCP': 'max'}).collect()[0]

In [112]: print(top['max(AVERAGE_PRCP)'])
4361.0
```

```
In [116]: daily_prcp.filter(F.col('AVERAGE_PRCP') == top['max(AVERAGE_PRCP)']).show()
+------------+----+------------+
|COUNTRY_CODE|YEAR|AVERAGE_PRCP|
+------------+----+------------+
|          EK|2000|      4361.0|
+------------+----+------------+
```

```
In [3]: df.where('COUNTRY_CODE == "EK"').show(1)
+-----------+----------+------------+--------+---------+---------+------+--------+--------+------+----------------+-----+-----
|         ID|STATE_CODE|COUNTRY_CODE|LATITUDE|LONGITUDE|ELEVATION|  NAME|GSN_FLAG|CRN_FLAG|WMO_ID|         COUNTRY|STATE|CORE_
+-----------+----------+------------+--------+---------+---------+------+--------+--------+------+----------------+-----+-----
|EKM00064810|          |          EK|   3.755|    8.709|     23.2|MALABO|        |        | 64810|Equatorial Guinea|     |
+-----------+----------+------------+--------+---------+---------+------+--------+--------+------+----------------+-----+-----
only showing top 1 row
```

Equatorial Guinea had the highest average rainfall in the year 2000, across the entire data set. Among the top 100 list this country appeared 4 times (1996,1997,2000 & 2001). But average precipitation values show high variations (4361,1100,709 & 576). These variations are suspectable.



I tried to plot cumulative rainfall using different file formats and libraries. But I was kept on getting the errors in one way or another. For some reason I could not get it displayed. I have submitted the code

Ancy John

Student ID: 52770710

that I worked in R. Since the file size for the geospatial shapefiles were very huge, I had to find out a way to subsetting the data. But all attempt failed.