```
import org.apache.spark.sql.SparkSession
val spark = SparkSession.builder
.appName("MongoDBSparkAnalysis")
.config("spark.mongodb.input.uri", "mongodb://localhost:27017/
weather_db.weather_collections")
.config("spark.mongodb.output.uri", "mongodb://localhost:27017/
weather_db.weather_collections")
 .getOrCreate()
val df = spark
 val spark = df.getOrCreate()
val weatherDF = spark.read
 .format("com.mongodb.spark.sql.DefaultSource")
 .option("uri", "mongodb://localhost:27017/
weather_db.weather_collections")
.load()
val weatherData = weatherDF.load()
 weatherData.printSchema()
```

```
val spark =
SparkSession.builder.appName("MongoDBSparkAnalysis").config("spark.m
ongodb.input.uri", "mongodb://localhost:27017/
weather_db.weather_collections").config("spark.mongodb.output.uri",
"mongodb://localhost:27017/
weather_db.weather_collections").getOrCreate()
```

```
Last login: Sat Dec 16 15:13:04 on ttys003
(base) mahanivethakannappan@Mahanivethas-MacBook-Air ~ % spark-shell
--packages org.mongodb.spark:mongo-spark-connector_2.12:3.0.1

23/12/16 15:16:13 WARN Utils: Your hostname, Mahanivethas-MacBook-
Air.local resolves to a loopback address: 127.0.0.1; using
192.168.29.67 instead (on interface en0)
23/12/16 15:16:13 WARN Utils: Set SPARK_LOCAL_IP if you need to bind
to another address
:: loading settings :: url = jar:file:/Users/mahanivethakannappan/
opt/anaconda3/lib/python3.9/site-packages/pyspark/jars/
ivy-2.5.1.jar!/org/apache/ivy/core/settings/ivysettings.xml
Ivy Default Cache set to: /Users/mahanivethakannappan/.ivy2/cache
The jars for the packages stored in: /Users/
mahanivethakannappan/.ivy2/jars
org.mongodb.spark#mongo-spark-connector_2.12 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-
parent-5ca10815-5d97-4892-8962-e74b2777da6d;1.0
    confs: [default]
    found org.mongodb.spark#mongo-spark-connector_2.12;3.0.1 in
central
    found org.mongodb#mongodb-driver-sync;4.0.5 in central
    found org.mongodb#bson;4.0.5 in central
    found org.mongodb#mongodb-driver-core;4.0.5 in central
```

```
:: resolution report :: resolve 991ms :: artifacts dl 37ms
    :: modules in use:
    org.mongodb#bson;4.0.5 from central in [default]
    org.mongodb#mongodb-driver-core;4.0.5 from central in [default]
    org.mongodb#mongodb-driver-sync;4.0.5 from central in [default]
    org.mongodb.spark#mongo-spark-connector_2.12;3.0.1 from central
in [default]
    ---------------------------------------------------------------
------
    |                  |            modules            ||
artifacts   |
    |       conf       | number| search|dwnlded|evicted|| number|
dwnlded|
    ---------------------------------------------------------------
------
    |      default     |   4   |   0   |   0   |   0   ||   4   |
0   |
    ---------------------------------------------------------------
------
:: retrieving :: org.apache.spark#spark-submit-
parent-5ca10815-5d97-4892-8962-e74b2777da6d
    confs: [default]
    0 artifacts copied, 4 already retrieved (0kB/31ms)
23/12/16 15:16:15 WARN NativeCodeLoader: Unable to load native-
hadoop library for your platform... using builtin-java classes where
applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR,
use setLogLevel(newLevel).
23/12/16 15:16:33 WARN Utils: Service 'SparkUI' could not bind on
port 4040. Attempting port 4041.
Spark context Web UI available at http://192.168.29.67:4041
Spark context available as 'sc' (master = local[*], app id =
local-1702719994069).
Spark session available as 'spark'.
23/12/16 15:16:44 WARN GarbageCollectionMetrics: To enable non-
built-in garbage collector(s) List(G1 Concurrent GC), users should
configure it(them) to
spark.eventLog.gcMetrics.youngGenerationGarbageCollectors or
spark.eventLog.gcMetrics.oldGenerationGarbageCollectors
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.5.0
      /_/

Using Scala version 2.12.18 (OpenJDK 64-Bit Server VM, Java 20)
Type in expressions to have them evaluated.
Type :help for more information.

scala> import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.SparkSession
```

```
scala>

val spark =
SparkSession.builder .appName("MongoDBSparkAnalysis") .config("spark
.mongodb.input.uri", "mongodb://
localhost:27017/ .config("spark.mongodb.output.uri", "mongodb://
localhost:27017/ .getOrCreate()

scala> val spark = SparkSession.builder
spark: org.apache.spark.sql.SparkSession.Builder =
org.apache.spark.sql.SparkSession$Builder@2be6103f

scala>   .appName("MongoDBSparkAnalysis")
res0: org.apache.spark.sql.SparkSession.Builder =
org.apache.spark.sql.SparkSession$Builder@2be6103f

scala>   .config("spark.mongodb.input.uri", "mongodb://
localhost:27017/weather_db.weather_collections")
res1: org.apache.spark.sql.SparkSession.Builder =
org.apache.spark.sql.SparkSession$Builder@2be6103f

scala>   .config("spark.mongodb.output.uri", "mongodb://
localhost:27017/weather_db.weather_collections")
res2: org.apache.spark.sql.SparkSession.Builder =
org.apache.spark.sql.SparkSession$Builder@2be6103f

scala>   .getOrCreate()
23/12/16 15:16:54 WARN SparkSession: Using an existing Spark
session; only runtime SQL configurations will take effect.
res3: org.apache.spark.sql.SparkSession =
org.apache.spark.sql.SparkSession@3c844476

scala>

scala> val df = spark
df: org.apache.spark.sql.SparkSession.Builder =
org.apache.spark.sql.SparkSession$Builder@2be6103f

scala> val spark = df.getOrCreate()
spark: org.apache.spark.sql.SparkSession =
org.apache.spark.sql.SparkSession@3c844476

scala> val weatherDF = spark.read
weatherDF: org.apache.spark.sql.DataFrameReader =
org.apache.spark.sql.DataFrameReader@5ed85a18

scala>   .format("com.mongodb.spark.sql.DefaultSource")
res4: org.apache.spark.sql.DataFrameReader =
org.apache.spark.sql.DataFrameReader@5ed85a18

scala>   .option("uri", "mongodb://localhost:27017/
weather_db.weather_collections")
res5: org.apache.spark.sql.DataFrameReader =
org.apache.spark.sql.DataFrameReader@5ed85a18
```

```
scala>   .load()
23/12/16 15:17:18 WARN SparkStringUtils: Truncated the string
representation of a plan since it was too large. This behavior can
be adjusted by setting 'spark.sql.debug.maxToStringFields'.
res6: org.apache.spark.sql.DataFrame = [_id: struct<oid: string>,
location: struct<lat: double, lon: double> ... 1 more field]

scala> val weatherData = weatherDF.load()
weatherData: org.apache.spark.sql.DataFrame = [_id: struct<oid:
string>, location: struct<lat: double, lon: double> ... 1 more
field]

scala> weatherData.printSchema()
root
 |-- _id: struct (nullable = true)
 |    |-- oid: string (nullable = true)
 |-- location: struct (nullable = true)
 |    |-- lat: double (nullable = true)
 |    |-- lon: double (nullable = true)
 |-- timelines: struct (nullable = true)
 |    |-- minutely: array (nullable = true)
 |    |    |-- element: struct (containsNull = true)
 |    |    |    |-- time: string (nullable = true)
 |    |    |    |-- values: struct (nullable = true)
 |    |    |    |    |-- cloudBase: double (nullable = true)
 |    |    |    |    |-- cloudCeiling: double (nullable = true)
 |    |    |    |    |-- cloudCover: double (nullable = true)
 |    |    |    |    |-- dewPoint: double (nullable = true)
 |    |    |    |    |-- freezingRainIntensity: integer (nullable =
true)
 |    |    |    |    |-- humidity: double (nullable = true)
 |    |    |    |    |-- precipitationProbability: integer (nullable
= true)
 |    |    |    |    |-- pressureSurfaceLevel: double (nullable =
true)
 |    |    |    |    |-- rainIntensity: integer (nullable = true)
 |    |    |    |    |-- sleetIntensity: integer (nullable = true)
 |    |    |    |    |-- snowIntensity: integer (nullable = true)
 |    |    |    |    |-- temperature: double (nullable = true)
 |    |    |    |    |-- temperatureApparent: double (nullable =
true)
 |    |    |    |    |-- uvHealthConcern: integer (nullable = true)
 |    |    |    |    |-- uvIndex: integer (nullable = true)
 |    |    |    |    |-- visibility: integer (nullable = true)
 |    |    |    |    |-- weatherCode: integer (nullable = true)
 |    |    |    |    |-- windDirection: double (nullable = true)
 |    |    |    |    |-- windGust: double (nullable = true)
 |    |    |    |    |-- windSpeed: double (nullable = true)
 |    |-- hourly: array (nullable = true)
 |    |    |-- element: struct (containsNull = true)
 |    |    |    |-- time: string (nullable = true)
 |    |    |    |-- values: struct (nullable = true)
 |    |    |    |    |-- cloudBase: double (nullable = true)
```

```
 |    |    |    |    |-- cloudCeiling: double (nullable = true)
 |    |    |    |    |-- cloudCover: double (nullable = true)
 |    |    |    |    |-- dewPoint: double (nullable = true)
 |    |    |    |    |-- evapotranspiration: double (nullable =
true)
 |    |    |    |    |-- freezingRainIntensity: integer (nullable =
true)
 |    |    |    |    |-- humidity: double (nullable = true)
 |    |    |    |    |-- iceAccumulation: integer (nullable = true)
 |    |    |    |    |-- iceAccumulationLwe: integer (nullable =
true)
 |    |    |    |    |-- precipitationProbability: integer (nullable
= true)
 |    |    |    |    |-- pressureSurfaceLevel: double (nullable =
true)
 |    |    |    |    |-- rainAccumulation: double (nullable = true)
 |    |    |    |    |-- rainAccumulationLwe: double (nullable =
true)
 |    |    |    |    |-- rainIntensity: double (nullable = true)
 |    |    |    |    |-- sleetAccumulation: integer (nullable =
true)
 |    |    |    |    |-- sleetAccumulationLwe: integer (nullable =
true)
 |    |    |    |    |-- sleetIntensity: integer (nullable = true)
 |    |    |    |    |-- snowAccumulation: integer (nullable = true)
 |    |    |    |    |-- snowAccumulationLwe: integer (nullable =
true)
 |    |    |    |    |-- snowIntensity: integer (nullable = true)
 |    |    |    |    |-- temperature: double (nullable = true)
 |    |    |    |    |-- temperatureApparent: double (nullable =
true)
 |    |    |    |    |-- uvHealthConcern: integer (nullable = true)
 |    |    |    |    |-- uvIndex: integer (nullable = true)
 |    |    |    |    |-- visibility: double (nullable = true)
 |    |    |    |    |-- weatherCode: integer (nullable = true)
 |    |    |    |    |-- windDirection: double (nullable = true)
 |    |    |    |    |-- windGust: double (nullable = true)
 |    |    |    |    |-- windSpeed: double (nullable = true)
 |    |-- daily: array (nullable = true)
 |    |    |-- element: struct (containsNull = true)
 |    |    |    |-- time: string (nullable = true)
 |    |    |    |-- values: struct (nullable = true)
 |    |    |    |    |-- cloudBaseAvg: double (nullable = true)
 |    |    |    |    |-- cloudBaseMax: double (nullable = true)
 |    |    |    |    |-- cloudBaseMin: double (nullable = true)
 |    |    |    |    |-- cloudCeilingAvg: double (nullable = true)
 |    |    |    |    |-- cloudCeilingMax: double (nullable = true)
 |    |    |    |    |-- cloudCeilingMin: integer (nullable = true)
 |    |    |    |    |-- cloudCoverAvg: double (nullable = true)
 |    |    |    |    |-- cloudCoverMax: double (nullable = true)
 |    |    |    |    |-- cloudCoverMin: double (nullable = true)
 |    |    |    |    |-- dewPointAvg: double (nullable = true)
 |    |    |    |    |-- dewPointMax: double (nullable = true)
 |    |    |    |    |-- dewPointMin: double (nullable = true)
```

```
 |    |    |    |    |-- evapotranspirationAvg: double (nullable =
true)
 |    |    |    |    |-- evapotranspirationMax: double (nullable =
true)
 |    |    |    |    |-- evapotranspirationMin: double (nullable =
true)
 |    |    |    |    |-- evapotranspirationSum: double (nullable =
true)
 |    |    |    |    |-- freezingRainIntensityAvg: integer (nullable
= true)
 |    |    |    |    |-- freezingRainIntensityMax: integer (nullable
= true)
 |    |    |    |    |-- freezingRainIntensityMin: integer (nullable
= true)
 |    |    |    |    |-- humidityAvg: double (nullable = true)
 |    |    |    |    |-- humidityMax: double (nullable = true)
 |    |    |    |    |-- humidityMin: double (nullable = true)
 |    |    |    |    |-- iceAccumulationAvg: integer (nullable =
true)
 |    |    |    |    |-- iceAccumulationLweAvg: integer (nullable =
true)
 |    |    |    |    |-- iceAccumulationLweMax: integer (nullable =
true)
 |    |    |    |    |-- iceAccumulationLweMin: integer (nullable =
true)
 |    |    |    |    |-- iceAccumulationLweSum: integer (nullable =
true)
 |    |    |    |    |-- iceAccumulationMax: integer (nullable =
true)
 |    |    |    |    |-- iceAccumulationMin: integer (nullable =
true)
 |    |    |    |    |-- iceAccumulationSum: integer (nullable =
true)
 |    |    |    |    |-- moonriseTime: string (nullable = true)
 |    |    |    |    |-- moonsetTime: string (nullable = true)
 |    |    |    |    |-- precipitationProbabilityAvg: double
(nullable = true)
 |    |    |    |    |-- precipitationProbabilityMax: integer
(nullable = true)
 |    |    |    |    |-- precipitationProbabilityMin: integer
(nullable = true)
 |    |    |    |    |-- pressureSurfaceLevelAvg: double (nullable =
true)
 |    |    |    |    |-- pressureSurfaceLevelMax: double (nullable =
true)
 |    |    |    |    |-- pressureSurfaceLevelMin: double (nullable =
true)
 |    |    |    |    |-- rainAccumulationAvg: double (nullable =
true)
 |    |    |    |    |-- rainAccumulationLweAvg: double (nullable =
true)
 |    |    |    |    |-- rainAccumulationLweMax: double (nullable =
true)
 |    |    |    |    |-- rainAccumulationLweMin: integer (nullable =
```

```
true)
 |    |    |    |    |-- rainAccumulationMax: double (nullable =
true)
 |    |    |    |    |-- rainAccumulationMin: integer (nullable =
true)
 |    |    |    |    |-- rainAccumulationSum: double (nullable =
true)
 |    |    |    |    |-- rainIntensityAvg: double (nullable = true)
 |    |    |    |    |-- rainIntensityMax: double (nullable = true)
 |    |    |    |    |-- rainIntensityMin: integer (nullable = true)
 |    |    |    |    |-- sleetAccumulationAvg: integer (nullable =
true)
 |    |    |    |    |-- sleetAccumulationLweAvg: integer (nullable
= true)
 |    |    |    |    |-- sleetAccumulationLweMax: integer (nullable
= true)
 |    |    |    |    |-- sleetAccumulationLweMin: integer (nullable
= true)
 |    |    |    |    |-- sleetAccumulationLweSum: integer (nullable
= true)
 |    |    |    |    |-- sleetAccumulationMax: integer (nullable =
true)
 |    |    |    |    |-- sleetAccumulationMin: integer (nullable =
true)
 |    |    |    |    |-- sleetIntensityAvg: integer (nullable =
true)
 |    |    |    |    |-- sleetIntensityMax: integer (nullable =
true)
 |    |    |    |    |-- sleetIntensityMin: integer (nullable =
true)
 |    |    |    |    |-- snowAccumulationAvg: integer (nullable =
true)
 |    |    |    |    |-- snowAccumulationLweAvg: integer (nullable =
true)
 |    |    |    |    |-- snowAccumulationLweMax: integer (nullable =
true)
 |    |    |    |    |-- snowAccumulationLweMin: integer (nullable =
true)
 |    |    |    |    |-- snowAccumulationLweSum: integer (nullable =
true)
 |    |    |    |    |-- snowAccumulationMax: integer (nullable =
true)
 |    |    |    |    |-- snowAccumulationMin: integer (nullable =
true)
 |    |    |    |    |-- snowAccumulationSum: integer (nullable =
true)
 |    |    |    |    |-- snowIntensityAvg: integer (nullable = true)
 |    |    |    |    |-- snowIntensityMax: integer (nullable = true)
 |    |    |    |    |-- snowIntensityMin: integer (nullable = true)
 |    |    |    |    |-- sunriseTime: string (nullable = true)
 |    |    |    |    |-- sunsetTime: string (nullable = true)
 |    |    |    |    |-- temperatureApparentAvg: double (nullable =
true)
 |    |    |    |    |-- temperatureApparentMax: double (nullable =
```

```
true)
 |    |    |    |    |-- temperatureApparentMin: double (nullable =
true)
 |    |    |    |    |-- temperatureAvg: double (nullable = true)
 |    |    |    |    |-- temperatureMax: double (nullable = true)
 |    |    |    |    |-- temperatureMin: double (nullable = true)
 |    |    |    |    |-- uvHealthConcernAvg: integer (nullable =
true)
 |    |    |    |    |-- uvHealthConcernMax: integer (nullable =
true)
 |    |    |    |    |-- uvHealthConcernMin: integer (nullable =
true)
 |    |    |    |    |-- uvIndexAvg: integer (nullable = true)
 |    |    |    |    |-- uvIndexMax: integer (nullable = true)
 |    |    |    |    |-- uvIndexMin: integer (nullable = true)
 |    |    |    |    |-- visibilityAvg: double (nullable = true)
 |    |    |    |    |-- visibilityMax: double (nullable = true)
 |    |    |    |    |-- visibilityMin: double (nullable = true)
 |    |    |    |    |-- weatherCodeMax: integer (nullable = true)
 |    |    |    |    |-- weatherCodeMin: integer (nullable = true)
 |    |    |    |    |-- windDirectionAvg: double (nullable = true)
 |    |    |    |    |-- windGustAvg: double (nullable = true)
 |    |    |    |    |-- windGustMax: double (nullable = true)
 |    |    |    |    |-- windGustMin: double (nullable = true)
 |    |    |    |    |-- windSpeedAvg: double (nullable = true)
 |    |    |    |    |-- windSpeedMax: double (nullable = true)
 |    |    |    |    |-- windSpeedMin: double (nullable = true)


scala> weatherData.select("timelines.daily.time",
"timelines.daily.values.temperatureAvg")
res8: org.apache.spark.sql.DataFrame = [time: array<string>,
temperatureAvg: array<double>]

scala>   .orderBy($"temperatureAvg".desc)
res9: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] =
[time: array<string>, temperatureAvg: array<double>]

scala>   .limit(1)
res10: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] =
[time: array<string>, temperatureAvg: array<double>]

scala>   .show()
+--------------------+--------------------+
|                time|      temperatureAvg|
+--------------------+--------------------+
|[2023-12-15T00:30...|[21.89, 22.27, 21...|
+--------------------+--------------------+


scala> import org.apache.spark.sql.functions.{explode, col}
import org.apache.spark.sql.functions.{explode, col}

scala>
```

```
scala> // Assuming your DataFrame is named 'weatherData'

scala> val explodedDF = weatherData.select(col("_id"),
     |   col("location"),
     |   explode(col("timelines.minutely")).as("minutely"))
explodedDF: org.apache.spark.sql.DataFrame = [_id: struct<oid:
string>, location: struct<lat: double, lon: double> ... 1 more
field]

scala>

scala> val formattedDF = explodedDF.select(
     |   col("_id"),
     |   col("location"),
     |   col("minutely.time").as("time"),
     |   col("minutely.values.cloudBase").as("cloudBase"),
     |   col("minutely.values.cloudCeiling").as("cloudCeiling"),
     |   col("minutely.values.cloudCover").as("cloudCover"),
     |   col("minutely.values.dewPoint").as("dewPoint"),
     |
col("minutely.values.freezingRainIntensity").as("freezingRainIntensi
ty"),
     |   col("minutely.values.humidity").as("humidity"),
     |
col("minutely.values.precipitationProbability").as("precipitationPro
bability"),
     |
col("minutely.values.pressureSurfaceLevel").as("pressureSurfaceLevel
"),
     |   col("minutely.values.rainIntensity").as("rainIntensity"),
     |   col("minutely.values.sleetIntensity").as("sleetIntensity"),
     |   col("minutely.values.snowIntensity").as("snowIntensity"),
     |   col("minutely.values.temperature").as("temperature"),
     |
col("minutely.values.temperatureApparent").as("temperatureApparent")
,
     |
col("minutely.values.uvHealthConcern").as("uvHealthConcern"),
     |   col("minutely.values.uvIndex").as("uvIndex"),
     |   col("minutely.values.visibility").as("visibility"),
     |   col("minutely.values.weatherCode").as("weatherCode"),
     |   col("minutely.values.windDirection").as("windDirection"),
     |   col("minutely.values.windGust").as("windGust"),
     |   col("minutely.values.windSpeed").as("windSpeed")
     | )
formattedDF: org.apache.spark.sql.DataFrame = [_id: struct<oid:
string>, location: struct<lat: double, lon: double> ... 21 more
fields]

scala>

scala> formattedDF.show(false)
```

```
+--------------------------+------------------+--------------------
+--------+-----------+---------+--------+--------------------
+--------+----------------------+-------------------
+------------+-------------+------------+----------
+------------------+--------------+-------+---------+----------
+------------+--------+---------+
|_id                       |location          |time                 |
cloudBase|cloudCeiling|cloudCover|dewPoint|freezingRainIntensity|
humidity|precipitationProbability|pressureSurfaceLevel|
rainIntensity|sleetIntensity|snowIntensity|temperature|
temperatureApparent|uvHealthConcern|uvIndex|visibility|weatherCode|
windDirection|windGust|windSpeed|
+--------------------------+------------------+--------------------
+--------+-----------+---------+--------+--------------------
+--------+----------------------+-------------------
+------------+-------------+------------+----------
+------------------+--------------+-------+---------+----------
+------------+--------+---------+
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T10:59:00Z|
0.55     |0.55        |75.0      |19.69   |0                    |
78.0    |0                        |910.81              |0
|0             |0            |25.81      |25.81              |0
|0      |16        |1001       |96.13        |3.19    |2.13     |
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:00:00Z|
1.07     |10.37       |99.22     |16.72   |0                    |
56.45   |0                        |910.81              |0
|0             |0            |26.04      |26.04              |0
|0      |16        |1001       |87.83        |7.64    |3.88     |
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:01:00Z|
1.07     |10.37       |99.23     |16.73   |0                    |
56.63   |0                        |910.82              |0
|0             |0            |26.01      |26.01              |0
|0      |16        |1001       |87.83        |7.64    |3.89     |
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:02:00Z|
1.07     |10.37       |99.24     |16.75   |0                    |
56.8    |0                        |910.83              |0
|0             |0            |25.98      |25.98              |0
|0      |16        |1001       |87.83        |7.65    |3.9      |
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:03:00Z|
1.07     |10.37       |99.26     |16.76   |0                    |
56.98   |0                        |910.84              |0
|0             |0            |25.95      |25.95              |0
|0      |16        |1001       |87.83        |7.65    |3.91     |
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:04:00Z|
1.07     |10.37       |99.27     |16.78   |0                    |
57.15   |0                        |910.85              |0
|0             |0            |25.92      |25.92              |0
|0      |16        |1001       |87.83        |7.65    |3.91     |
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:05:00Z|
1.07     |10.37       |99.28     |16.79   |0                    |
57.32   |0                        |910.86              |0
```

|0 |0 |25.89 |25.89 |0
|0 |16 |1001 |87.83 |7.65 |3.92 |
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:06:00Z|
1.07 |10.37 |99.3 |16.81 |0 |
57.5 |0 |910.87 |0
|0 |0 |25.85 |25.85 |0
|0 |16 |1001 |87.83 |7.66 |3.93 |
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:07:00Z|
1.07 |10.37 |99.31 |16.82 |0 |
57.67 |0 |910.88 |0
|0 |0 |25.82 |25.82 |0
|0 |16 |1001 |87.83 |7.66 |3.94 |
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:08:00Z|
1.07 |10.37 |99.32 |16.84 |0 |
57.85 |0 |910.89 |0
|0 |0 |25.79 |25.79 |0
|0 |16 |1001 |87.83 |7.66 |3.94 |
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:09:00Z|
1.07 |10.37 |99.34 |16.85 |0 |
58.02 |0 |910.9 |0
|0 |0 |25.76 |25.76 |0
|0 |16 |1001 |87.83 |7.66 |3.95 |
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:10:00Z|
1.07 |10.37 |99.35 |16.87 |0 |
58.19 |0 |910.91 |0
|0 |0 |25.73 |25.73 |0
|0 |16 |1001 |87.83 |7.66 |3.96 |
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:11:00Z|
1.07 |10.37 |99.36 |16.88 |0 |
58.37 |0 |910.92 |0
|0 |0 |25.7 |25.7 |0
|0 |16 |1001 |87.83 |7.67 |3.97 |
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:12:00Z|
1.07 |10.37 |99.38 |16.9 |0 |
58.54 |0 |910.93 |0
|0 |0 |25.66 |25.66 |0
|0 |16 |1001 |87.83 |7.67 |3.97 |
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:13:00Z|
1.07 |10.37 |99.39 |16.91 |0 |
58.71 |0 |910.93 |0
|0 |0 |25.63 |25.63 |0
|0 |16 |1001 |87.83 |7.67 |3.98 |
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:14:00Z|
1.07 |10.37 |99.4 |16.93 |0 |
58.89 |0 |910.94 |0
|0 |0 |25.6 |25.6 |0
|0 |16 |1001 |87.83 |7.67 |3.99 |
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:15:00Z|
1.07 |10.37 |99.41 |16.94 |0 |
59.06 |0 |910.95 |0
|0 |0 |25.57 |25.57 |0
|0 |16 |1001 |87.83 |7.67 |4.0 |
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:16:00Z|
1.07 |10.37 |99.43 |16.96 |0 |

```
59.24   |0                        |910.96               |0
|0              |0            |25.54      |25.54              |0
|0      |16        |1001       |87.83        |7.68    |4.0      |
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:17:00Z|
1.07     |10.37       |99.44     |16.97   |0                      |
59.41   |0                        |910.97               |0
|0              |0            |25.51      |25.51              |0
|0      |16        |1001       |87.83        |7.68    |4.01     |
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:18:00Z|
1.07     |10.37       |99.45     |16.99   |0                      |
59.58   |0                        |910.98               |0
|0              |0            |25.48      |25.48              |0
|0      |16        |1001       |87.83        |7.68    |4.02     |
+-------------------------+-----------------+-------------------
+--------+-----------+---------+--------+-------------------
+--------+----------------------+-------------------
+-------------+--------------+-------------+-----------
+------------------+--------------+-------+----------+----------
+-------------+--------+---------+
only showing top 20 rows
```

```
_____
scala> import org.apache.spark.sql.functions._
import org.apache.spark.sql.functions._

scala>

scala> // Explode the minutely array

scala> val minutelyDF = flattenedDF.selectExpr("_id", "location",
"explode(timelines.minutely.time) as minutely_time")
minutelyDF: org.apache.spark.sql.DataFrame = [_id: struct<oid:
string>, location: struct<lat: double, lon: double> ... 1 more
field]

scala>

scala> // Explode the hourly array

scala> val hourlyDF = flattenedDF.selectExpr("_id", "location",
"explode(timelines.hourly.time) as hourly_time")
hourlyDF: org.apache.spark.sql.DataFrame = [_id: struct<oid:
string>, location: struct<lat: double, lon: double> ... 1 more
field]

scala>

scala> // Explode the daily array

scala> val dailyDF = flattenedDF.selectExpr("_id", "location",
"explode(timelines.daily.time) as daily_time")
```

```
dailyDF: org.apache.spark.sql.DataFrame = [_id: struct<oid: string>,
location: struct<lat: double, lon: double> ... 1 more field]

scala>

scala> // Perform joins to combine the results

scala> val resultDF = minutelyDF
resultDF: org.apache.spark.sql.DataFrame = [_id: struct<oid:
string>, location: struct<lat: double, lon: double> ... 1 more
field]

scala>   .join(hourlyDF, Seq("_id", "location"), "outer")
res36: org.apache.spark.sql.DataFrame = [_id: struct<oid: string>,
location: struct<lat: double, lon: double> ... 2 more fields]

scala>   .join(dailyDF, Seq("_id", "location"), "outer")
res37: org.apache.spark.sql.DataFrame = [_id: struct<oid: string>,
location: struct<lat: double, lon: double> ... 3 more fields]

scala>

scala> // Show the resulting DataFrame

scala> resultDF.show()
+--------------------+------------------+--------------------+
|                 _id|          location|       minutely_time|
+--------------------+------------------+--------------------+
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-15T10:59:00Z|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-15T11:00:00Z|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-15T11:01:00Z|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-15T11:02:00Z|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-15T11:03:00Z|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-15T11:04:00Z|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-15T11:05:00Z|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-15T11:06:00Z|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-15T11:07:00Z|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-15T11:08:00Z|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-15T11:09:00Z|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-15T11:10:00Z|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-15T11:11:00Z|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-15T11:12:00Z|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-15T11:13:00Z|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-15T11:14:00Z|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-15T11:15:00Z|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-15T11:16:00Z|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-15T11:17:00Z|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-15T11:18:00Z|
+--------------------+------------------+--------------------+
only showing top 20 rows

scala> import org.apache.spark.sql.functions._
import org.apache.spark.sql.functions._
```

```
scala>

scala> val resultDF = flattenedDF.select(
     |   col("_id"),
     |   col("location"),
     |   expr("timelines.minutely.time[0]").as("minutely_time"),
     |   expr("timelines.hourly.time[0]").as("hourly_time"),
     |   expr("timelines.daily.time[0]").as("daily_time")
     | )
resultDF: org.apache.spark.sql.DataFrame = [_id: struct<oid:
string>, location: struct<lat: double, lon: double> ... 3 more
fields]

scala>

scala> resultDF.show()
+--------------------+-----------------+--------------------
+--------------------+--------------------+
|                 _id|         location|       minutely_time|
hourly_time|          daily_time|
+--------------------+-----------------+--------------------
+--------------------+--------------------+
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-15T10:59:00Z|
2023-12-15T10:00:00Z|2023-12-15T00:30:00Z|
+--------------------+-----------------+--------------------
+--------------------+--------------------+
scala> val resultDF = flattenedDF.select(
     |   col("_id"),
     |   col("location"),
     |   expr("timelines.minutely.time[10]").as("minutely_time"),
     |   expr("timelines.hourly.time[10]").as("hourly_time"),
     |   expr("timelines.daily.time[10]").as("daily_time")
     | )
resultDF: org.apache.spark.sql.DataFrame = [_id: struct<oid:
string>, location: struct<lat: double, lon: double> ... 3 more
fields]

scala>

scala> resultDF.show()
+--------------------+-----------------+--------------------
+--------------------+----------+
|                 _id|         location|       minutely_time|
hourly_time|daily_time|
+--------------------+-----------------+--------------------
+--------------------+----------+
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-15T11:09:00Z|
2023-12-15T20:00:00Z|      NULL|
+--------------------+-----------------+--------------------
+--------------------+----------+
```

Corrected

_____

```
scala> val explodedMinuteliesDF = resultDF.selectExpr(
     |   "_id",
     |   "location",
     |   "explode(timelines.minutely.time) as minutely_time"
     | )
explodedMinuteliesDF: org.apache.spark.sql.DataFrame = [_id:
struct<oid: string>, location: struct<lat: double, lon: double> ...
1 more field]

scala>

scala> val explodedHourliesDF = resultDF.selectExpr(
     |   "_id",
     |   "location",
     |   "explode(timelines.hourly.time) as hourly_time"
     | )
explodedHourliesDF: org.apache.spark.sql.DataFrame = [_id:
struct<oid: string>, location: struct<lat: double, lon: double> ...
1 more field]

scala>

scala> val explodedDailiesDF = resultDF.selectExpr(
     |   "_id",
     |   "location",
     |   "explode(timelines.daily.time) as daily_time"
     | )
explodedDailiesDF: org.apache.spark.sql.DataFrame = [_id:
struct<oid: string>, location: struct<lat: double, lon: double> ...
1 more field]

scala> explodedMinuteliesDF.show(truncate = false)
+-------------------------+-----------------+--------------------+
|_id                      |location         |minutely_time       |
+-------------------------+-----------------+--------------------+
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T10:59:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:00:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:01:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:02:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:03:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:04:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:05:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:06:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:07:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:08:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:09:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:10:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:11:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:12:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:13:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:14:00Z|
```

```
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:15:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:16:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:17:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:18:00Z|
+--------------------------+------------------+--------------------+
only showing top 20 rows


scala> explodedHourliesDF.show(truncate = false)
+--------------------------+------------------+--------------------+
|_id                       |location          |hourly_time         |
+--------------------------+------------------+--------------------+
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T10:00:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T11:00:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T12:00:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T13:00:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T14:00:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T15:00:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T16:00:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T17:00:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T18:00:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T19:00:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T20:00:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T21:00:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T22:00:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T23:00:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-16T00:00:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-16T01:00:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-16T02:00:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-16T03:00:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-16T04:00:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-16T05:00:00Z|
+--------------------------+------------------+--------------------+
only showing top 20 rows


scala> explodedDailiesDF.show(truncate = false)
+--------------------------+------------------+--------------------+
|_id                       |location          |daily_time          |
+--------------------------+------------------+--------------------+
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-15T00:30:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-16T00:30:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-17T00:30:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-18T00:30:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-19T00:30:00Z|
|{657c31b2b8dc5929a778b295}|{12.9716, 77.5946}|2023-12-20T00:30:00Z|
+--------------------------+------------------+--------------------+


_________
Moonrise moon set
_____________
scala> val explodedTimelinesDF = resultDF.selectExpr(
     |   "_id",
```

```
     |   "location",
     |   "explode(timelines.daily) as daily_data"
     | )
explodedTimelinesDF: org.apache.spark.sql.DataFrame = [_id:
struct<oid: string>, location: struct<lat: double, lon: double> ...
1 more field]

scala>

scala> val extractedMoonTimesDF = explodedTimelinesDF.select(
     |   "_id",
     |   "location",
     |   "daily_data.time",
     |   "daily_data.values.moonriseTime",
     |   "daily_data.values.moonsetTime"
     | )
extractedMoonTimesDF: org.apache.spark.sql.DataFrame = [_id:
struct<oid: string>, location: struct<lat: double, lon: double> ...
3 more fields]

scala>

scala> // Now, let's calculate the difference in seconds

scala> val diffDF = extractedMoonTimesDF.selectExpr(
     |   "_id",
     |   "location",
     |   "time",
     |   "moonriseTime",
     |   "moonsetTime",
     |   "unix_timestamp(moonsetTime, 'yyyy-MM-dd HH:mm:ss') -
unix_timestamp(moonriseTime, 'yyyy-MM-dd HH:mm:ss') as difference"
     | )
diffDF: org.apache.spark.sql.DataFrame = [_id: struct<oid: string>,
location: struct<lat: double, lon: double> ... 4 more fields]


scala> diffDF.show()
+------------------+-----------------+------------------
+------------------+------------------+----------+
|               _id|         location|              time|
moonriseTime|        moonsetTime|difference|
+------------------+-----------------+------------------
+------------------+------------------+----------+
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-15T00:30:00Z|
2023-12-15T03:10:22Z|2023-12-15T14:48:48Z|      NULL|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-16T00:30:00Z|
2023-12-16T04:07:05Z|2023-12-16T15:52:15Z|      NULL|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-17T00:30:00Z|
2023-12-17T04:58:55Z|2023-12-17T16:53:50Z|      NULL|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-18T00:30:00Z|
2023-12-18T05:47:33Z|2023-12-18T17:52:30Z|      NULL|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-19T00:30:00Z|
2023-12-19T06:32:23Z|2023-12-19T18:49:27Z|      NULL|
```

```
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-20T00:30:00Z|
2023-12-20T07:14:35Z|2023-12-20T19:44:56Z|      NULL|
+--------------------+------------------+--------------------
+--------------------+--------------------+----------+


______________
 Calculate daily summaries
______________


scala> import org.apache.spark.sql.functions._
import org.apache.spark.sql.functions._

scala> val explodedDailyDF = weatherData.selectExpr(
     |   "_id",
     |   "location",
     |   "explode(timelines.daily) as daily_data"
     | )
explodedDailyDF: org.apache.spark.sql.DataFrame = [_id: struct<oid:
string>, location: struct<lat: double, lon: double> ... 1 more
field]

scala>

scala> // Extract relevant columns

scala> val extractedDailyMetricsDF = explodedDailyDF.select(
     |   "_id",
     |   "location",
     |   "daily_data.time",
     |   "daily_data.values.temperatureAvg",
     |   "daily_data.values.temperatureMax",
     |   "daily_data.values.temperatureMin",
     |   "daily_data.values.humidityAvg",
     |   "daily_data.values.humidityMax",
     |   "daily_data.values.humidityMin"
     |   // Add other relevant columns here
     | )
extractedDailyMetricsDF: org.apache.spark.sql.DataFrame = [_id:
struct<oid: string>, location: struct<lat: double, lon: double> ...
7 more fields]

scala> // Calculate daily summaries

scala> val dailySummariesDF = extractedDailyMetricsDF.groupBy(
     |   "_id",
     |   "location",
     |   "time" // Use the correct column name "time"
     | ).agg(
     |   avg("temperatureAvg").alias("avgTemperature"),
     |   max("temperatureMax").alias("maxTemperature"),
     |   min("temperatureMin").alias("minTemperature"),
```

```
     |   avg("humidityAvg").alias("avgHumidity"),
     |   max("humidityMax").alias("maxHumidity"),
     |   min("humidityMin").alias("minHumidity")
     |   // Add other aggregation functions for additional metrics
     | )
dailySummariesDF: org.apache.spark.sql.DataFrame = [_id: struct<oid:
string>, location: struct<lat: double, lon: double> ... 7 more
fields]

scala>

scala> // Show the resulting DataFrame

scala> dailySummariesDF.show()
+--------------------+-----------------+--------------------
+-------------+-------------+-------------+-----------
+-----------+-----------+
|                 _id|         location|                time|
avgTemperature|maxTemperature|minTemperature|avgHumidity|
maxHumidity|minHumidity|
+--------------------+-----------------+--------------------
+-------------+-------------+-------------+-----------
+-----------+-----------+
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-16T00:30:00Z|
22.27|         27.26|         18.31|      78.09|      92.17|
54.3|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-20T00:30:00Z|
22.53|         25.87|         17.02|      55.59|      82.41|
44.15|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-18T00:30:00Z|
20.87|         26.27|         16.86|      69.49|      95.82|
43.89|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-17T00:30:00Z|
21.5|         27.58|         17.04|      76.54|       96.5|
47.36|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-15T00:30:00Z|
21.89|         26.63|         18.38|      84.45|       97.0|
56.45|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-19T00:30:00Z|
21.1|         26.14|          17.3|      58.95|      81.36|
29.72|
+--------------------+-----------------+--------------------
+-------------+-------------+-------------+-----------
+-----------+-----------+
_______
```

Extreme Weather Events:

```
 scala> import org.apache.spark.sql.functions._
import org.apache.spark.sql.functions._

scala> import org.apache.spark.sql.SparkSession
```

```
import org.apache.spark.sql.SparkSession

scala>

scala> // Create a Spark session

scala> val spark =
SparkSession.builder.appName("WeatherAnalysis").getOrCreate()
23/12/16 23:20:53 WARN SparkSession: Using an existing Spark
session; only runtime SQL configurations will take effect.
spark: org.apache.spark.sql.SparkSession =
org.apache.spark.sql.SparkSession@382b008d

scala>

scala> // Assuming you have loaded the weather data into a DataFrame
named 'weatherData'

scala> // Please replace "temperatureMax" with the actual column
name containing maximum temperature

scala>

scala> // Select relevant columns

scala> val temperatureData = weatherData.select(
     |   col("_id"),
     |   col("location"),
     |   explode(col("timelines.daily")).as("daily_data")
     | ).select(
     |   col("_id"),
     |   col("location"),
     |   col("daily_data.time"),
     |
col("daily_data.values.temperatureMax").as("maxTemperature")
     | )
temperatureData: org.apache.spark.sql.DataFrame = [_id: struct<oid:
string>, location: struct<lat: double, lon: double> ... 2 more
fields]

scala>
scala> // Adjusted threshold

scala> val newExtremeTemperatureThreshold = 25.0
newExtremeTemperatureThreshold: Double = 25.0

scala>

scala> // Identify days with extreme temperature

scala> val newExtremeTemperatureDays =
temperatureData.filter(col("maxTemperature") >
newExtremeTemperatureThreshold)
newExtremeTemperatureDays:
```

```
org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [_id:
struct<oid: string>, location: struct<lat: double, lon: double> ...
2 more fields]

scala>

scala> // Show the resulting DataFrame

scala> newExtremeTemperatureDays.show()
+--------------------+-----------------+--------------------
+-------------+
|                 _id|         location|                time|
maxTemperature|
+--------------------+-----------------+--------------------
+-------------+
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-15T00:30:00Z|
26.63|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-16T00:30:00Z|
27.26|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-17T00:30:00Z|
27.58|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-18T00:30:00Z|
26.27|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-19T00:30:00Z|
26.14|
|{657c31b2b8dc5929...|{12.9716, 77.5946}|2023-12-20T00:30:00Z|
25.87|
+--------------------+-----------------+--------------------
+-------------+

__________
CLOUD COVER
_________
scala> import org.apache.spark.sql.functions.col
import org.apache.spark.sql.functions.col

scala>

scala> val cloudCoverAnalysisDF = weatherData.select(
     |   col("timelines.minutely.time").alias("time"),
     |
col("timelines.minutely.values.cloudCover").alias("cloudCover"),
     |
col("timelines.minutely.values.temperature").alias("temperature"),
     |
col("timelines.minutely.values.humidity").alias("humidity"),
     |
col("timelines.minutely.values.precipitationProbability").alias("pre
cipitationProbability")
     | )
cloudCoverAnalysisDF: org.apache.spark.sql.DataFrame = [time:
array<string>, cloudCover: array<double> ... 3 more fields]

scala> import org.apache.spark.sql.functions.{col, expr}
```

```
import org.apache.spark.sql.functions.{col, expr}

scala>

scala> val cloudCoverAnalysisDF = weatherData.select(
     |   expr("timelines.minutely.time[0]").alias("time"),
     |
expr("timelines.minutely.values.cloudCover[0]").alias("cloudCover"),
     |
expr("timelines.minutely.values.temperature[0]").alias("temperature"
),
     |
expr("timelines.minutely.values.humidity[0]").alias("humidity"),
     |
expr("timelines.minutely.values.precipitationProbability[0]").alias(
"precipitationProbability")
     | )
cloudCoverAnalysisDF: org.apache.spark.sql.DataFrame = [time:
string, cloudCover: double ... 3 more fields]

scala> cloudCoverAnalysisDF.show()
+--------------------+---------+-----------+--------
+------------------------+
|                time|cloudCover|temperature|humidity|
precipitationProbability|
+--------------------+---------+-----------+--------
+------------------------+
|2023-12-15T10:59:00Z|      75.0|      25.81|    78.0|
0|
+--------------------+---------+-----------+--------
+------------------------+




==================
Weather code
==================
scala> val maxWeatherCode =
weatherData.orderBy(desc("timelines.minutely.values.precipitationPro
bability")).first()
maxWeatherCode: org.apache.spark.sql.Row =
[[657c31b2b8dc5929a778b295],[12.9716,77.5946],
[WrappedArray([2023-12-15T10:59:00Z,
[0.55,0.55,75.0,19.69,0,78.0,0,910.81,0,0,0,25.81,25.81,0,0,16,1001,
96.13,3.19,2.13]], [2023-12-15T11:00:00Z,
[1.07,10.37,99.22,16.72,0,56.45,0,910.81,0,0,0,26.04,26.04,0,0,16,10
01,87.83,7.64,3.88]], [2023-12-15T11:01:00Z,
[1.07,10.37,99.23,16.73,0,56.63,0,910.82,0,0,0,26.01,26.01,0,0,16,10
01,87.83,7.64,3.89]], [2023-12-15T11:02:00Z,
[1.07,10.37,99.24,16.75,0,56.8,0,910.83,0,0,0,25.98,25.98,0,0,16,100
1,87.83,7.65,3.9]], [2023-12-15T11:03:00Z,
[1.07,10.37,99.26,16.76,0,56.98,0,910.84,0,0,0,25.95,25.95,0,0,16,10
01,87.83,7.65,3.91]], [2023-12-15T11:04:00Z,
```

```
[1.07,10.37,99.27,16.78,0,57.15,0,910.85,0,0,0,25.92,25.92,0,0,16,10
01,87.83,7.65,3.91]], [...

scala>

scala> // Code to find minWeatherCode

scala> val minWeatherCode =
weatherData.orderBy("timelines.minutely.values.precipitationProbabil
ity").first()
minWeatherCode: org.apache.spark.sql.Row =
[[657c31b2b8dc5929a778b295],[12.9716,77.5946],
[WrappedArray([2023-12-15T10:59:00Z,
[0.55,0.55,75.0,19.69,0,78.0,0,910.81,0,0,0,25.81,25.81,0,0,16,1001,
96.13,3.19,2.13]], [2023-12-15T11:00:00Z,
[1.07,10.37,99.22,16.72,0,56.45,0,910.81,0,0,0,26.04,26.04,0,0,16,10
01,87.83,7.64,3.88]], [2023-12-15T11:01:00Z,
[1.07,10.37,99.23,16.73,0,56.63,0,910.82,0,0,0,26.01,26.01,0,0,16,10
01,87.83,7.64,3.89]], [2023-12-15T11:02:00Z,
[1.07,10.37,99.24,16.75,0,56.8,0,910.83,0,0,0,25.98,25.98,0,0,16,100
1,87.83,7.65,3.9]], [2023-12-15T11:03:00Z,
[1.07,10.37,99.26,16.76,0,56.98,0,910.84,0,0,0,25.95,25.95,0,0,16,10
01,87.83,7.65,3.91]], [2023-12-15T11:04:00Z,
[1.07,10.37,99.27,16.78,0,57.15,0,910.85,0,0,0,25.92,25.92,0,0,16,10
01,87.83,7.65,3.91]], [...

scala>

scala> // Display the content of maxWeatherCode

scala> val columnNamesMax = maxWeatherCode.schema.fieldNames
columnNamesMax: Array[String] = Array(_id, location, timelines)

scala> val valuesMax = maxWeatherCode.toSeq
valuesMax: Seq[Any] = WrappedArray([657c31b2b8dc5929a778b295],
[12.9716,77.5946], [WrappedArray([2023-12-15T10:59:00Z,
[0.55,0.55,75.0,19.69,0,78.0,0,910.81,0,0,0,25.81,25.81,0,0,16,1001,
96.13,3.19,2.13]], [2023-12-15T11:00:00Z,
[1.07,10.37,99.22,16.72,0,56.45,0,910.81,0,0,0,26.04,26.04,0,0,16,10
01,87.83,7.64,3.88]], [2023-12-15T11:01:00Z,
[1.07,10.37,99.23,16.73,0,56.63,0,910.82,0,0,0,26.01,26.01,0,0,16,10
01,87.83,7.64,3.89]], [2023-12-15T11:02:00Z,
[1.07,10.37,99.24,16.75,0,56.8,0,910.83,0,0,0,25.98,25.98,0,0,16,100
1,87.83,7.65,3.9]], [2023-12-15T11:03:00Z,
[1.07,10.37,99.26,16.76,0,56.98,0,910.84,0,0,0,25.95,25.95,0,0,16,10
01,87.83,7.65,3.91]], [2023-12-15T11:04:00Z,
[1.07,10.37,99.27,16.78,0,57.15,0,910.85,0,0,0,25.92,25.92,0,0,16,10
01,87.83,7.65,3.91]], [2023-12...

=========
Ml
=======
scala> import org.apache.spark.ml.regression.RandomForestRegressor
import org.apache.spark.ml.regression.RandomForestRegressor
```

```
scala>

scala> // Assuming df is your DataFrame

scala>

scala> // Extracting features from the nested structure

scala> val extractedData = weatherData.select(
     |   "location.lat",
     |   "location.lon",
     |   "timelines.minutely.values.cloudBase",
     |   "timelines.minutely.values.cloudCover",
     |   "timelines.minutely.values.dewPoint",
     |   // Add more features as needed
     | )
extractedData: org.apache.spark.sql.DataFrame = [lat: double, lon:
double ... 3 more fields]

scala>

scala> // Renaming columns for simplicity

scala> val renamedData = extractedData
renamedData: org.apache.spark.sql.DataFrame = [lat: double, lon:
double ... 3 more fields]

scala>   .withColumnRenamed("lat", "latitude")
res390: org.apache.spark.sql.DataFrame = [latitude: double, lon:
double ... 3 more fields]

scala>   .withColumnRenamed("lon", "longitude")
res391: org.apache.spark.sql.DataFrame = [latitude: double,
longitude: double ... 3 more fields]

scala>   .withColumnRenamed("cloudBase", "cloudBaseMinutely")
res392: org.apache.spark.sql.DataFrame = [latitude: double,
longitude: double ... 3 more fields]

scala>   .withColumnRenamed("cloudCover", "cloudCoverMinutely")
res393: org.apache.spark.sql.DataFrame = [latitude: double,
longitude: double ... 3 more fields]

scala>   .withColumnRenamed("dewPoint", "dewPointMinutely")
res394: org.apache.spark.sql.DataFrame = [latitude: double,
longitude: double ... 3 more fields]

scala>   // Rename more columns as needed

scala>

scala> // Assemble features into a single vector column
```

```
scala> val featureCols = Array("latitude", "longitude",
"cloudBaseMinutely", "cloudCoverMinutely", "dewPointMinutely")
featureCols: Array[String] = Array(latitude, longitude,
cloudBaseMinutely, cloudCoverMinutely, dewPointMinutely)

scala> val assembler = new VectorAssembler()
assembler: org.apache.spark.ml.feature.VectorAssembler =
VectorAssembler: uid=vecAssembler_0cd4819df27a, handleInvalid=error

scala>   .setInputCols(featureCols)
res395: assembler.type = VectorAssembler:
uid=vecAssembler_0cd4819df27a, handleInvalid=error, numInputCols=5

scala>   .setOutputCol("features")
res396: res395.type = VectorAssembler:
uid=vecAssembler_0cd4819df27a, handleInvalid=error, numInputCols=5

scala>

scala> // Create a RandomForestRegressor

scala> val rf = new RandomForestRegressor()
rf: org.apache.spark.ml.regression.RandomForestRegressor =
rfr_411001424378

scala>   .setLabelCol("your_target_column")  // Replace with your
actual target column
res397: org.apache.spark.ml.regression.RandomForestRegressor =
rfr_411001424378

scala>   .setFeaturesCol("features")
res398: org.apache.spark.ml.regression.RandomForestRegressor =
rfr_411001424378

scala>

scala> // Create a pipeline

scala> val pipeline = new Pipeline()
pipeline: org.apache.spark.ml.Pipeline = pipeline_a0c066d4a33b

scala>   .setStages(Array(assembler, rf))
res399: pipeline.type = pipeline_a0c066d4a33b

scala>

scala> val Array(trainingData, testData) =
renamedData.randomSplit(Array(0.8, 0.2))
trainingData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row]
= [lat: double, lon: double ... 3 more fields]
testData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] =
[lat: double, lon: double ... 3 more fields]
```

```
// Train the model
val model = pipeline.fit(trainingData)

// Make predictions on the test set
val predictions = model.transform(testData)

// Evaluate your model (you may need to replace RegressionEvaluator
with a suitable evaluator for your task)
val evaluator = new RegressionEvaluator()
  .setLabelCol("your_target_column")
  .setPredictionCol("prediction")
val rmse = evaluator.evaluate(predictions)

// Print the RMSE (Root Mean Squared Error)
println(s"Root Mean Squared Error (RMSE) on test data: $rmse")
```

SVD

```
scala> import org.apache.spark.mllib.linalg.distributed.RowMatrix
import org.apache.spark.mllib.linalg.distributed.RowMatrix

scala> import org.apache.spark.mllib.linalg.{Matrix,
SingularValueDecomposition, Vectors}
import org.apache.spark.mllib.linalg.{Matrix,
SingularValueDecomposition, Vectors}

scala> import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.SparkSession

scala>

scala> // Create a Spark session

scala> val spark =
SparkSession.builder.appName("SVDExample").getOrCreate()
spark: org.apache.spark.sql.SparkSession =
org.apache.spark.sql.SparkSession@446de37d

scala>
```

```
scala> // Sample data (you should replace this with your own data)

scala> val data = Seq(
     |   Vectors.dense(1.0, 2.0, 3.0),
     |   Vectors.dense(4.0, 5.0, 6.0),
     |   Vectors.dense(7.0, 8.0, 9.0)
     | )
data: Seq[org.apache.spark.mllib.linalg.Vector] =
List([1.0,2.0,3.0], [4.0,5.0,6.0], [7.0,8.0,9.0])

scala>

scala> // Create an RDD of vectors

scala> val rows = spark.sparkContext.parallelize(data)
rows: org.apache.spark.rdd.RDD[org.apache.spark.mllib.linalg.Vector]
= ParallelCollectionRDD[13] at parallelize at <console>:44

scala>

scala> // Create a RowMatrix

scala> val rowMatrix = new RowMatrix(rows)
rowMatrix: org.apache.spark.mllib.linalg.distributed.RowMatrix =
org.apache.spark.mllib.linalg.distributed.RowMatrix@29f47269

scala>

scala> // Compute SVD

scala> val svd: SingularValueDecomposition[RowMatrix, Matrix] =
rowMatrix.computeSVD(3, computeU = true)
svd:
org.apache.spark.mllib.linalg.SingularValueDecomposition[org.apache.
spark.mllib.linalg.distributed.RowMatrix,org.apache.spark.mllib.lina
lg.Matrix] =
SingularValueDecomposition(org.apache.spark.mllib.linalg.distributed
.RowMatrix@6af02ef3,
[16.848103352614206,1.0683695145547043,8.527732601206366E-8],-0.4796
711778777719  0.7766909903215564   0.4082482904638688
-0.5723677939720624  0.07568647010456442  -0.8164965809277254
-0.6650644100663532  -0.6253180501124457  0.4082482904638585   )

scala>

scala> // U, S, and V are the result of SVD

scala> val U: RowMatrix = svd.U
U: org.apache.spark.mllib.linalg.distributed.RowMatrix =
org.apache.spark.mllib.linalg.distributed.RowMatrix@6af02ef3

scala> val s: org.apache.spark.mllib.linalg.Vector = svd.s
s: org.apache.spark.mllib.linalg.Vector =
[16.848103352614206,1.0683695145547043,8.527732601206366E-8]
```

```
scala> val V: Matrix = svd.V
V: org.apache.spark.mllib.linalg.Matrix =
-0.4796711778777719  0.7766909903215564   0.4082482904638688
-0.5723677939720624  0.07568647010456442  -0.8164965809277254
-0.6650644100663532  -0.6253180501124457  0.4082482904638585

scala>

scala> // Print the results

scala> println("U:")
U:

scala> U.rows.foreach(println)
[-0.8263375405610782,0.38794278236977675,5.960464477539063E-8]
[-0.2148372383683963 5,-0.8872306883463738,-7.636845111846924E-8]
[-0.5205873894647374,-0.24964395298829833,-7.450580596923828E-9]

scala>

scala> println("Singular values:")
Singular values:

scala> s.toArray.foreach(println)
16.848103352614206
1.0683695145547043
8.527732601206366E-8

scala>

scala> println("V:")
V:

scala> println(V)
-0.4796711778777719  0.7766909903215564   0.4082482904638688
-0.5723677939720624  0.07568647010456442  -0.8164965809277254
-0.6650644100663532  -0.6253180501124457  0.4082482904638585

scala>
```