COURSEWORK: Academic Year 2025/2026

**CMM705 Big Data Programming**

Anjana Dodampe

**RGU ID: 2522702**

**IIT ID: 20250693**



**Table of Contents**

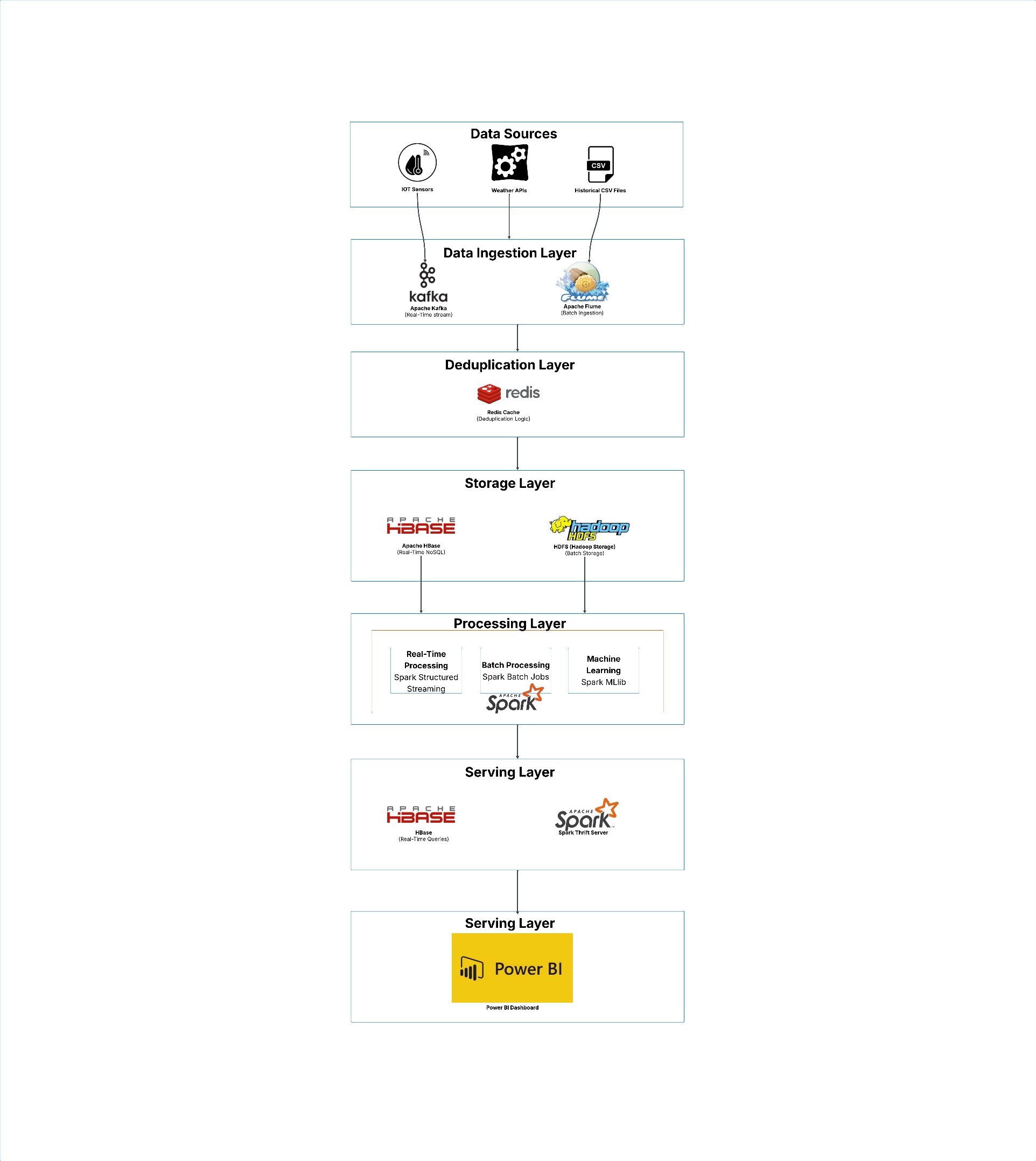
[**[Task 1] Designing a Solution Architecture**](#Task_1)

[**[Task 2] Data Analysis**](#Task_2)

[**[Task 3] Performing Machine Learning model using Spark MLlib**](#Task_3)

[**[Task 4] Presentation of the Analysis**](#Task_4)

**[Task 1] - Designing a Solution Architecture**

[Task 1] [1] Solution Architecture Design

**[Task 1] [2] Roles of the components of the Designed Solution Architecture**

1. Data Ingestion Layer

**Apache Kafka** → Collects live weather data

* Weather stations send data every 10-15 minutes
* Stores messages for 7 days (backup if processing fails)
* Partitions data by district (parallel processing)

**Apache Flume** → Loads historical CSV files

* Monitors folder for new CSV files
* Automatically uploads to HDFS
* Runs hourly batch ingestion

2. Deduplication Layer

**Redis Cache** → Prevents duplicate records

* Creates unique key: district:timestamp:temperature
* Check if data has already been seen in last 2 hours
* If duplicate → discard, if new → pass to storage
* Saves 30-40% storage by removing duplicates

3. Storage Layer

**HBase** → Fast access to recent data (last 30 days)

* Stores latest readings for real-time dashboard
* Query speed: <10ms (instant dashboard updates)
* Auto-deletes old data after 30 days (stays lean)

**HDFS** → Long-term historical storage (15+ years)

* Keeps all raw and processed data
* Scales to petabytes (add more nodes as needed)
* 3x replication (survives hardware failures)
* Uses Parquet format (10x smaller than CSV)

4. Processing Layer

**Apache Spark** → Does all the work (one tool for everything)

* Spark Streaming: Processes live Kafka data every 10 seconds
  + Reads messages → deduplicates → writes to HBase + HDFS
* Spark Batch: Scheduled jobs for aggregations
  + Daily: Calculate average temp, total rainfall per district
  + Hourly: Clean data, remove outliers
* Spark MLlib: Machine learning predictions
  + Trains model to predict evapotranspiration (ET₀)
  + Help farmers know how much to irrigate

5. Serving Layer

**HBase REST API** → Dashboard queries latest data

* When a user through Power BI asks "What's Colombo's current temperature?"
* HBase responds in <10ms with live data

**Spark Thrift Server** → Historical SQL queries

* When a user through Power BI asks "What was average rainfall in 2023?"
* Spark SQL queries HDFS and returns results
* Analysts run custom SQL queries

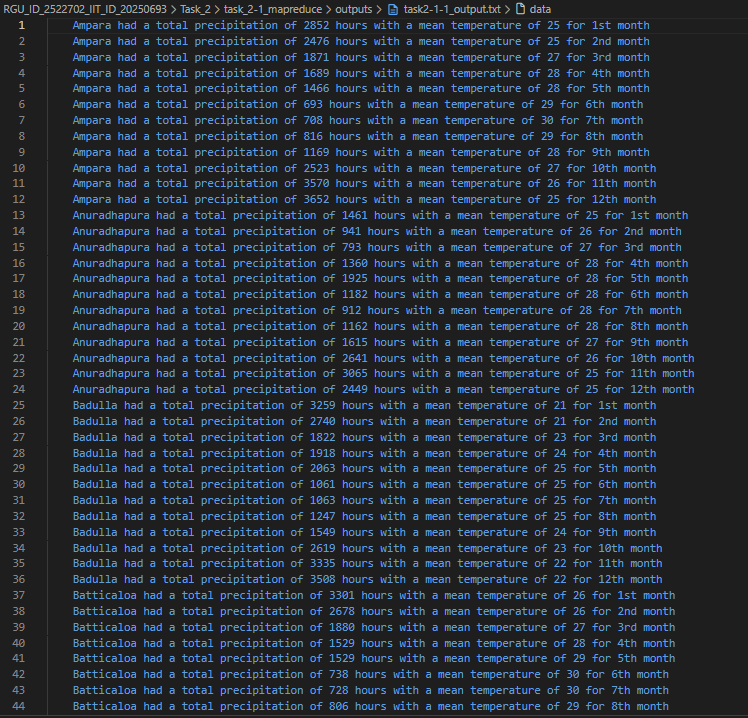
6. Visualization Layer

**Power BI Dashboard** → User interface

* Real-time panel: Shows live weather (refreshes every 30 sec)
* Historical trends: Charts showing patterns over years
* ML predictions: Evapotranspiration forecasts for irrigation planning

**[Task 2] Data Analysis**

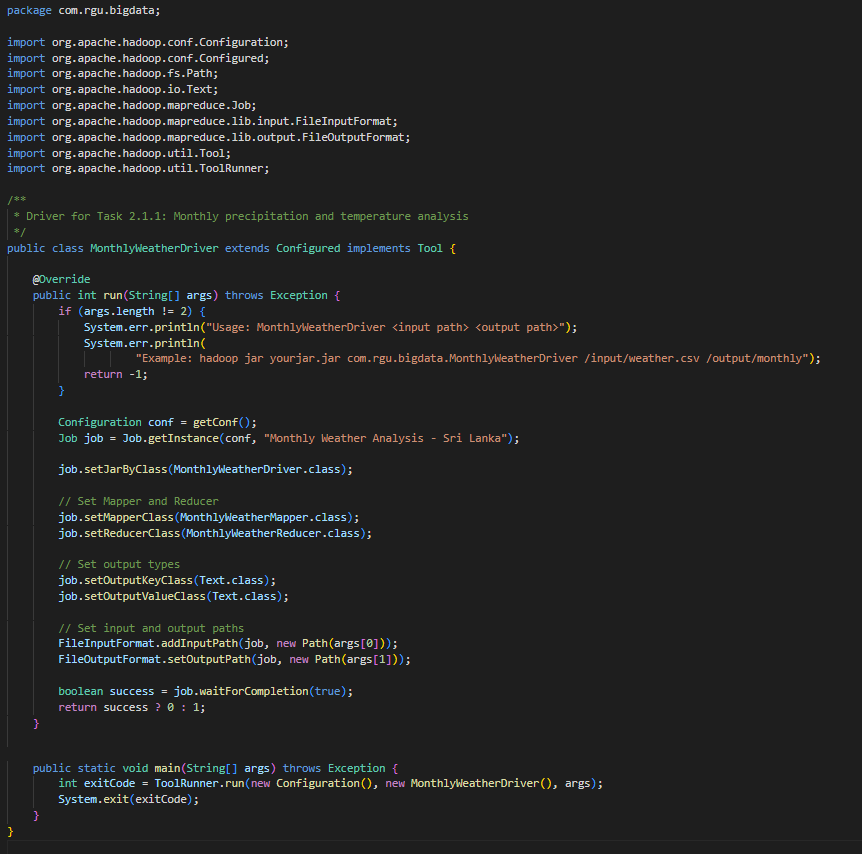
**Task 2 [1] 1 – Hadoop MapReduce** **Analysis - Final Output:**

****

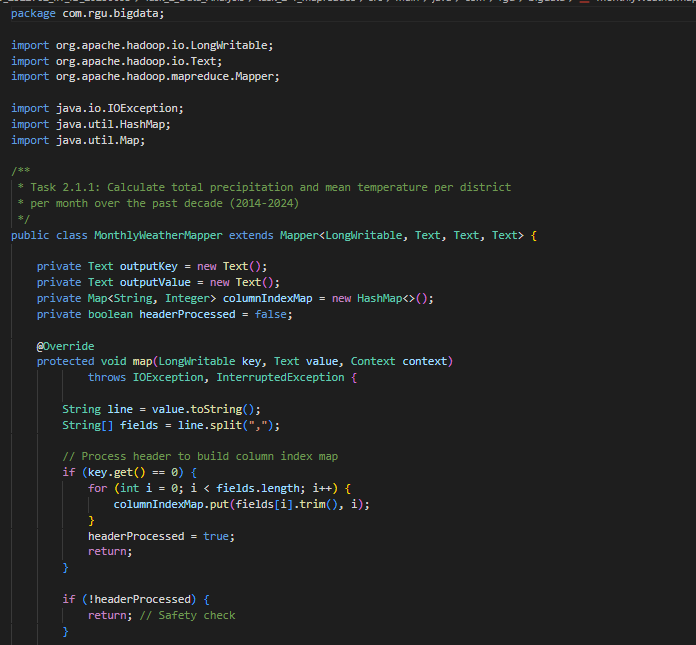
**Task 2 [1] 1 - Implementation:**

I have implemented the following Java classes.

MonthlyWeatherDriver.java - Configures and launches the job

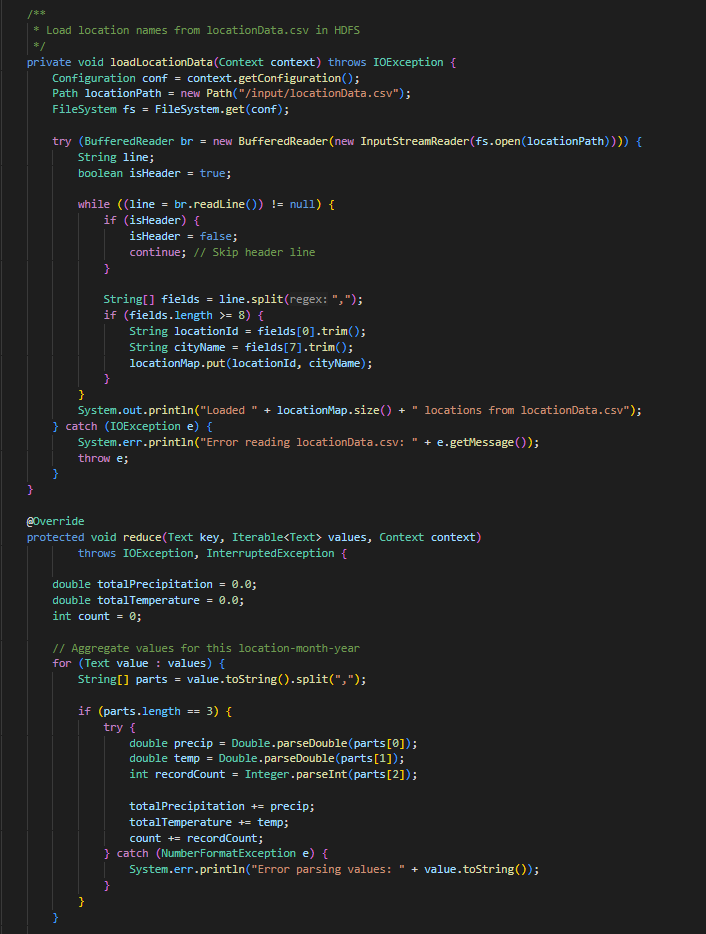
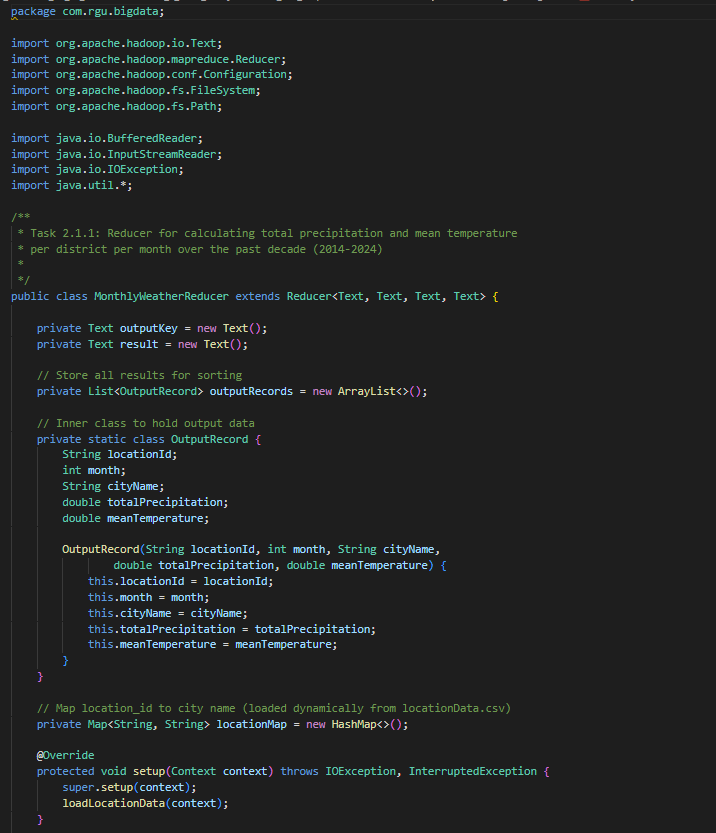


MonthlyWeatherMapper.java - Extracts location, month, year, precipitation, temperature

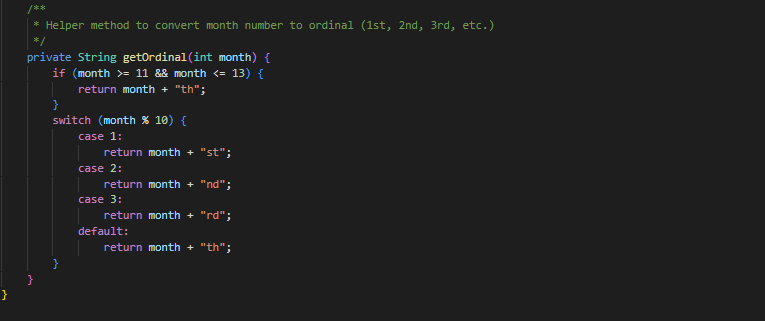




MonthlyWeatherReducer.java - Aggregates data per district, per month

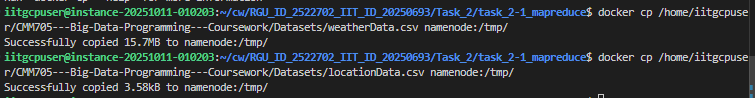




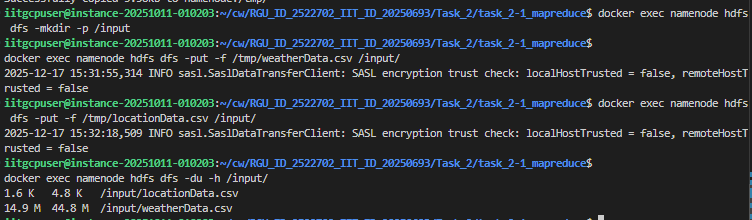


**Task 2 [1] 1 – Execution Steps:**

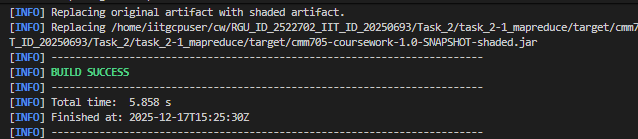
1. Copied datasets to namenode container.



2. Uploaded copied data into HDFS



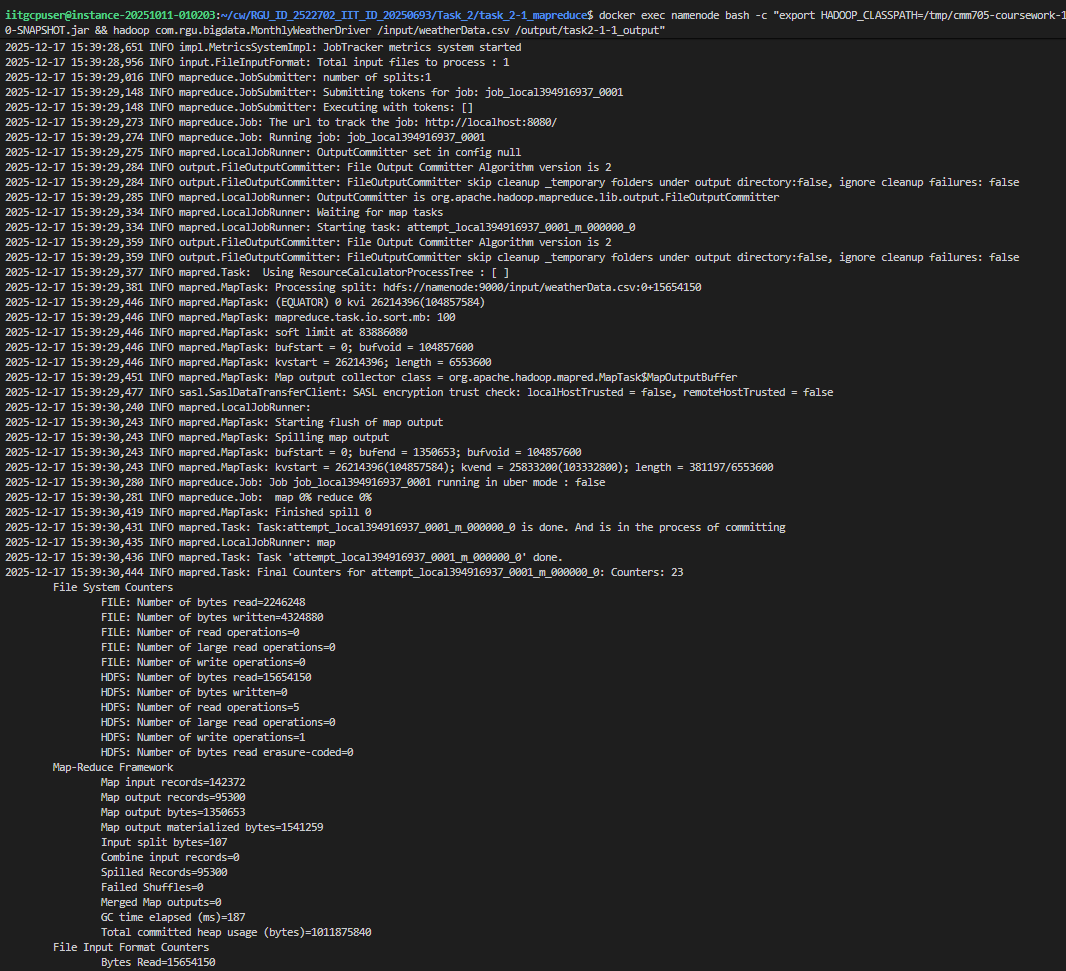
3. Built the MapReduce JAR

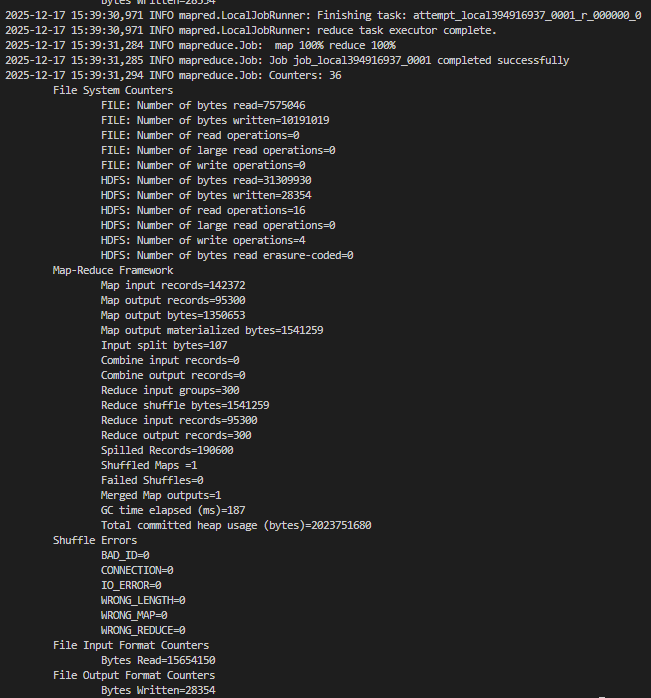


4. Copied the generated JAR into namenode.



5. Ran MapReduce Job MonthlyWeatherDriver.

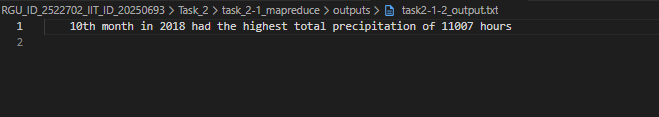




* MonthlyWeatherDriver job has executed successfully, and results had saved into a txt file in HDFS.

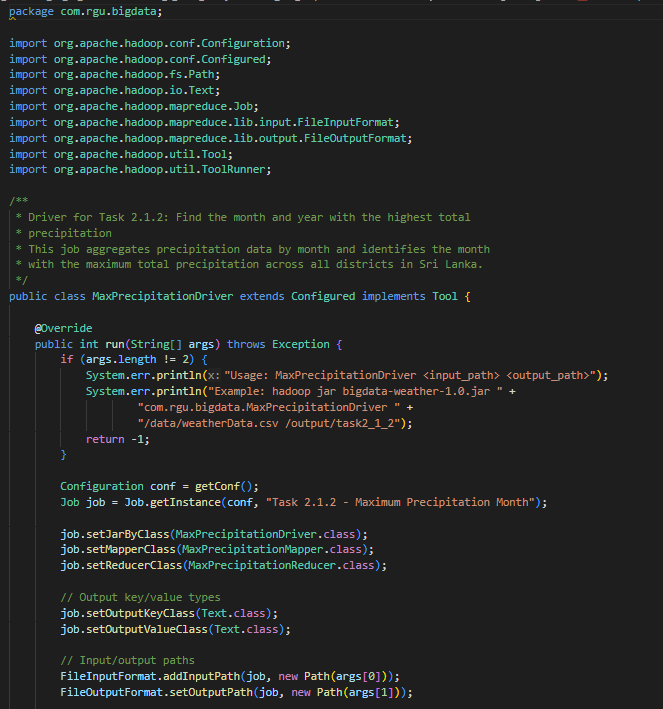
**[Task 2] 1. 2 Hadoop MapReduce Analysis**

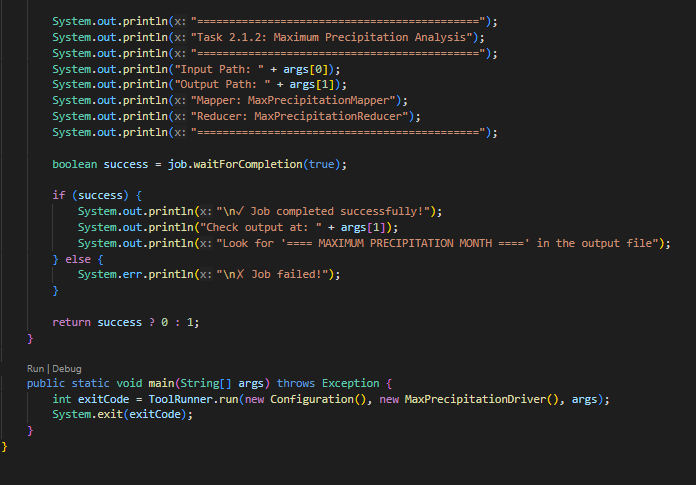
**Task 2 [1] 2 – Hadoop MapReduce** **Analysis - Final Output:**

****

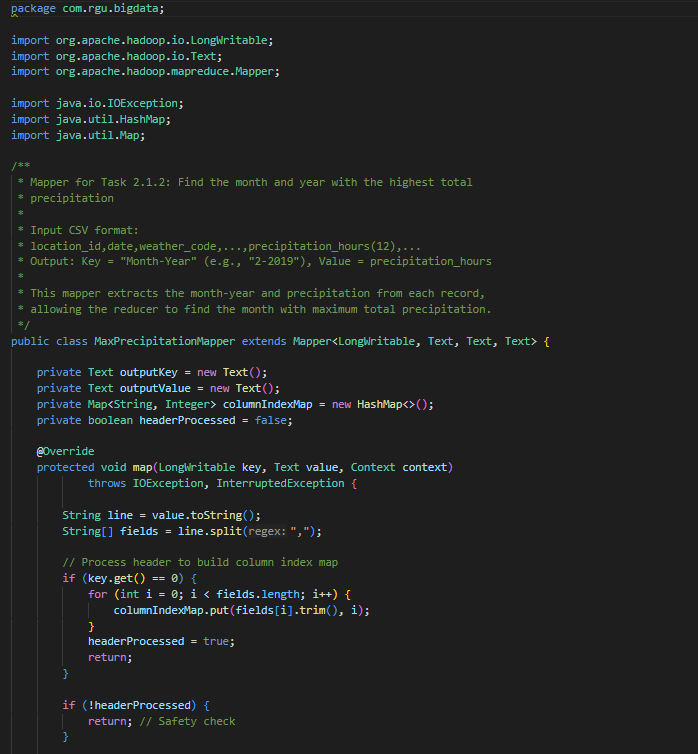
**Task 2 [1] 2 – Implementation:**

* MaxPrecipitationDriver.java - Configures and launches the job



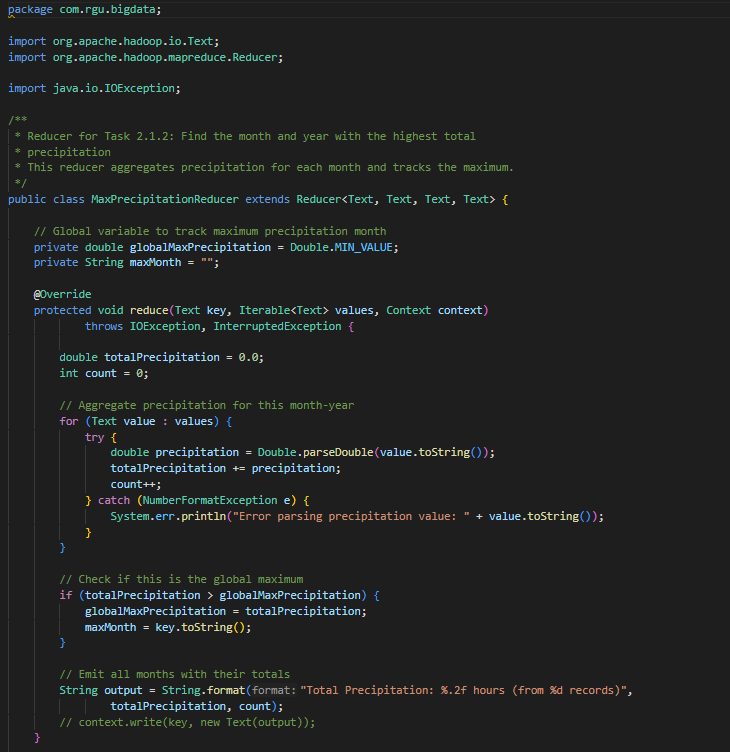


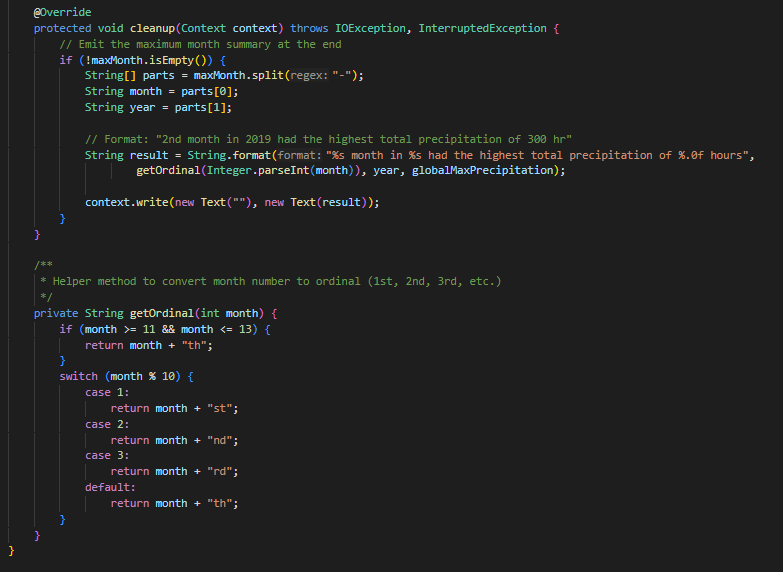
* MaxPrecipitationMapper.java - Extracts month, year, precipitation





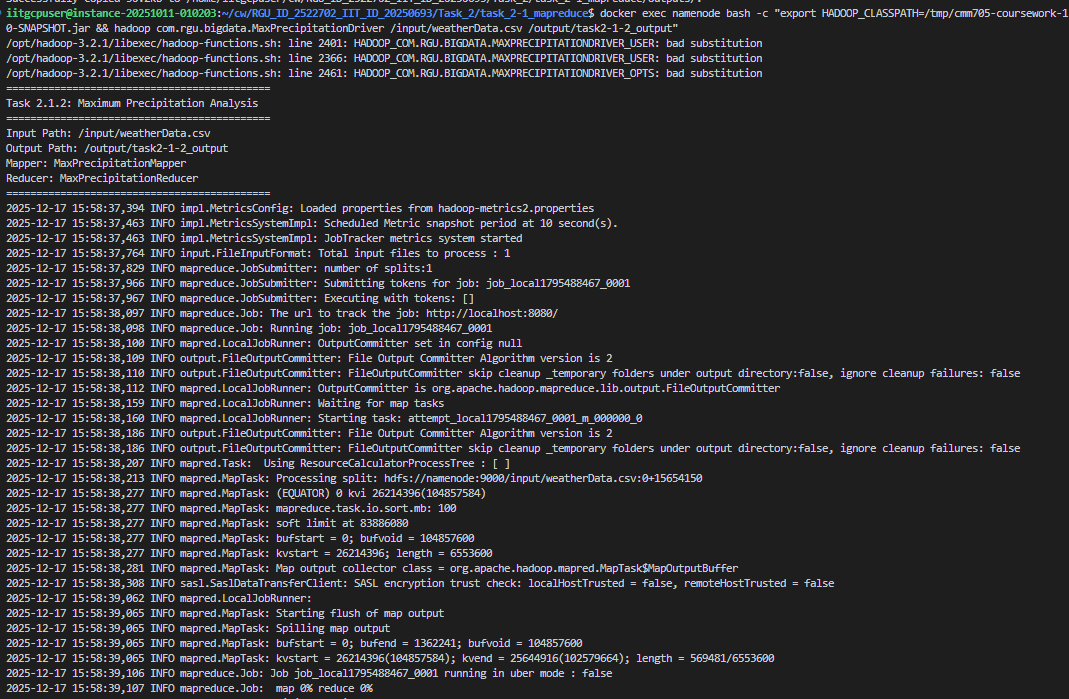
* MaxPrecipitationReducer.java - Finds month with highest precipitation

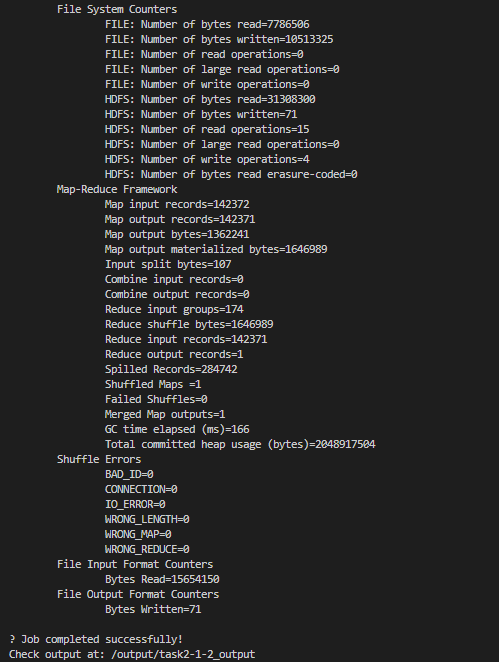




**Task 2 [1] 2 – Execution Steps:**

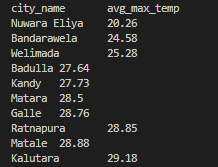
1. After copying the input datasets into HDFS as in previous step, executed the MapReduce MaxPrecipitationDriver job.



****

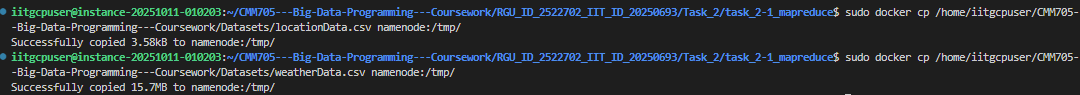
[Task 2] 2. Hive Analysis

**Task 2 [2] 1 – Hive Analysis - Final Output:**

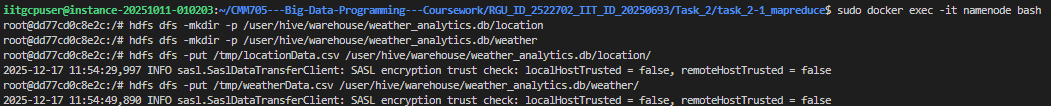


**Task 2 [2] 1 - Implementation steps:**

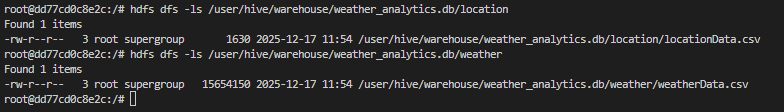
1. Copied Datasets to namenode.



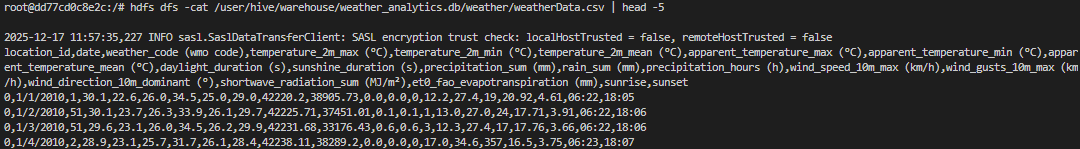
2. Entered the Hadoop namenode container, created directories and uploaded the copied datasets to HDFS.



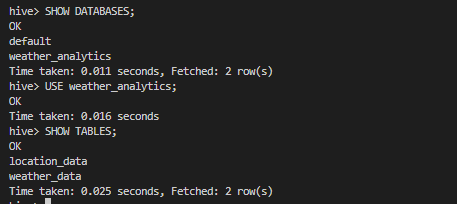
Data had uploaded correctly to HDFS.



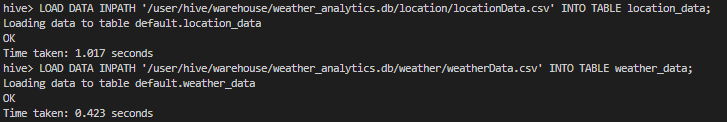


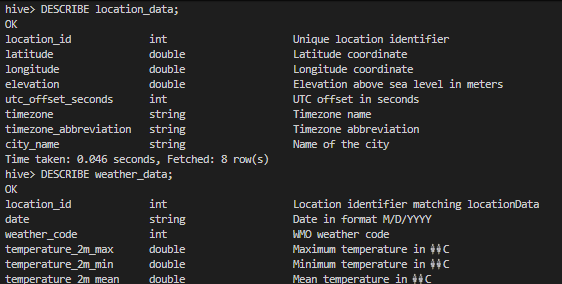


3. Connected to Hive server and created a Database and Tables in Hive.

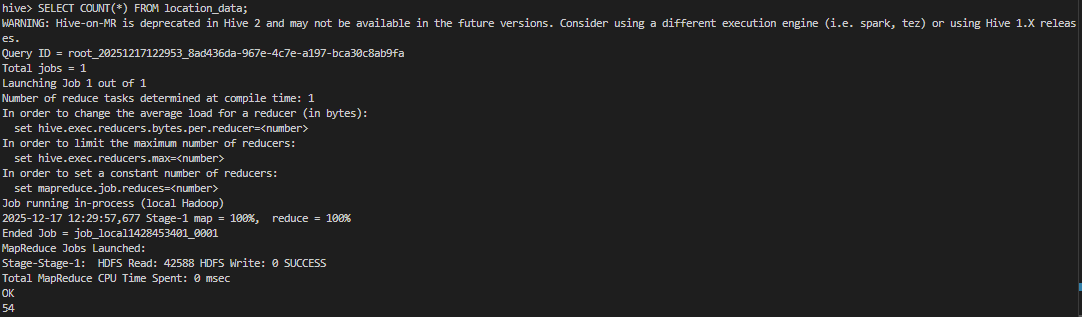


4. Loaded data in HDFS into the tables created.





Successfully loaded data could be seen in both tables.





5. Executed the following Hive query to Rank the top 10 most temperate cities across the dataset.

SELECT

l.city\_name AS city\_name,

ROUND(AVG(w.temperature\_2m\_max), 2) AS avg\_max\_temp

FROM

weather\_data w

JOIN

location\_data l

ON w.location\_id = l.location\_id

WHERE

w.temperature\_2m\_max IS NOT NULL

GROUP BY

l.city\_name

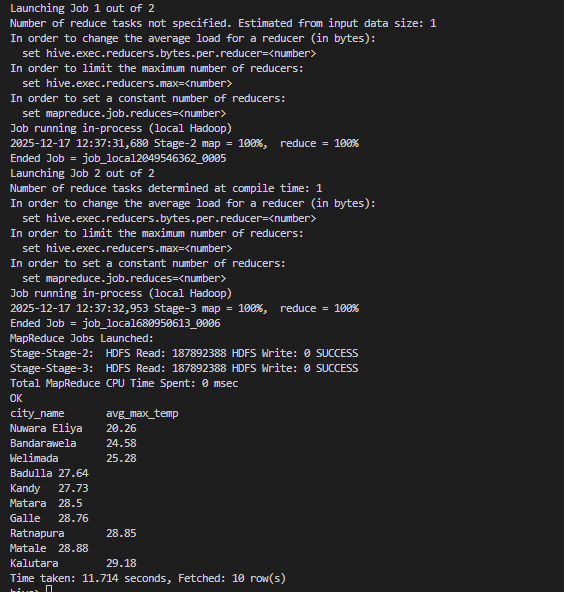
ORDER BY

avg\_max\_temp ASC

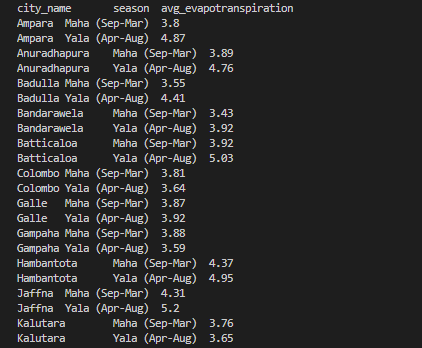
LIMIT 10;

The following configuration was set in order to display column names in the output.

*set hive.cli.print.header=true;*



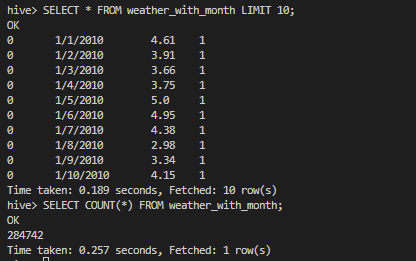
**Task 2 [2] 1 – Hive Analysis - Final Output:**

****

**Task 2 [2] 1 - Implementation steps:**

Followed the same steps from Step 1 to 4 as in above task.

1. Since the Date format in the provided dataset is M/D/YYYY or MM/DD/YYYY, first I created a temporary table with parsed dates. We can use this to extract month from the date strig.



2. Executed the following Hive query to calculate the average evapotranspiration for each major agricultural season in each district over the years.

SELECT

l.city\_name AS city\_name,

CASE

WHEN w.month\_num IN (9, 10, 11, 12, 1, 2, 3) THEN 'Maha (Sep-Mar)'

WHEN w.month\_num IN (4, 5, 6, 7, 8) THEN 'Yala (Apr-Aug)'

END AS season,

ROUND(AVG(w.et0\_fao\_evapotranspiration), 2) AS avg\_evapotranspiration

FROM

weather\_with\_month w

JOIN

location\_data l

ON w.location\_id = l.location\_id

WHERE

w.month\_num IN (1,2,3,4,5,6,7,8,9,10,11,12)

GROUP BY

l.city\_name,

CASE

WHEN w.month\_num IN (9, 10, 11, 12, 1, 2, 3) THEN 'Maha (Sep-Mar)'

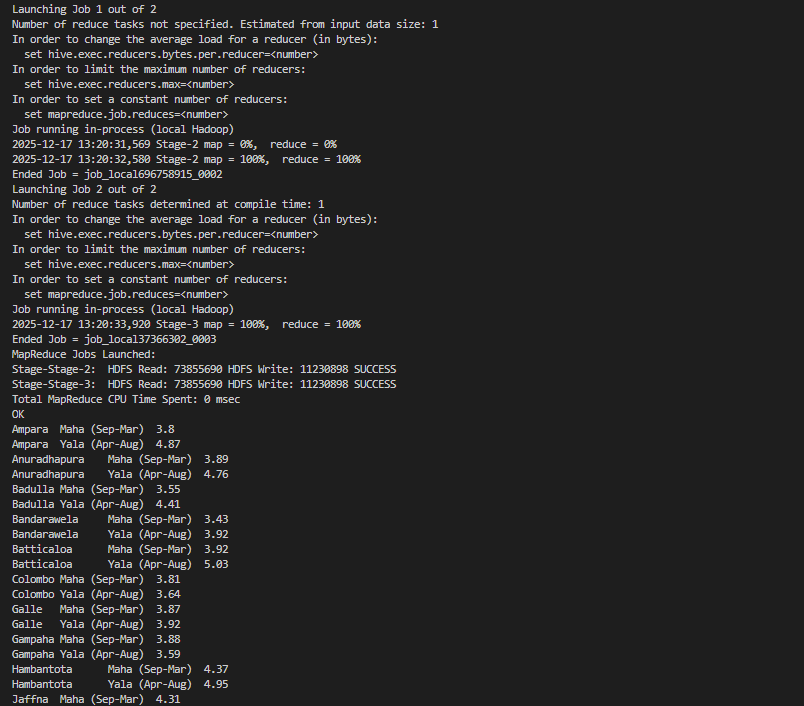
WHEN w.month\_num IN (4, 5, 6, 7, 8) THEN 'Yala (Apr-Aug)'

END

ORDER BY

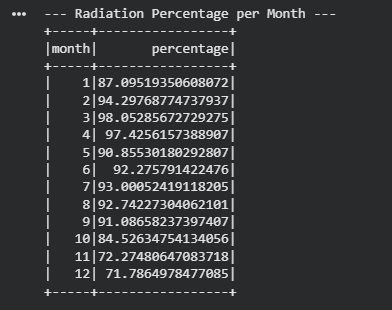
city\_name,

season;



[Task 2] 3. Spark Analysis

**Task 2 [3] 1 – Spark Analysis - Final Output:**

****

**Task 2 [3] 1 - Implementation:**

Pre-processed data to be used in spark analysis.

from pyspark.sql import SparkSession

from pyspark.sql import functions as F

from pyspark.sql.window import Window

# Initialize Spark Session

spark = SparkSession.builder.appName("SL\_Weather\_Analysis").getOrCreate()

# Load Datasets

loc\_df = spark.read.csv("/content/drive/MyDrive/CW\_DATASETS/locationData.csv", header=True, inferSchema=True)

weather\_df = spark.read.csv("/content/drive/MyDrive/CW\_DATASETS/weatherData.csv", header=True, inferSchema=True)

# Preprocessing

# Convert date string to DateType (Format in CSV is M/d/yyyy based on inspection)

weather\_df = weather\_df.withColumn("date", F.to\_date(F.col("date"), "M/d/yyyy"))

weather\_df = weather\_df.withColumn("year", F.year("date")) \

                       .withColumn("month", F.month("date"))

# Join with Location Data to get City Names

df = weather\_df.join(loc\_df, "location\_id")

# Cache df for better performance on multiple queries

df.cache()

I mainly focused on quantifying how much of the monthly shortwave radiation exceeded a threshold of **15 MJ/m²** across all districts.

* Sum of shortwave radiation values was calculated for each month.
* Radiation values greater than 15 MJ/m² were isolated and summed separately.
* The ratio of high radiation to total radiation was computed for each month.
* This percentage represents the share of monthly radiation that surpassed the 15 MJ/m² threshold.

# Calculate Total Radiation and High Radiation (>15) per month

q1\_result = df.groupBy("month").agg(

    F.sum("shortwave\_radiation\_sum (MJ/m²)").alias("total\_radiation"),

    F.sum(F.when(F.col("shortwave\_radiation\_sum (MJ/m²)") > 15,

                 F.col("shortwave\_radiation\_sum (MJ/m²)")).otherwise(0)).alias("high\_radiation")

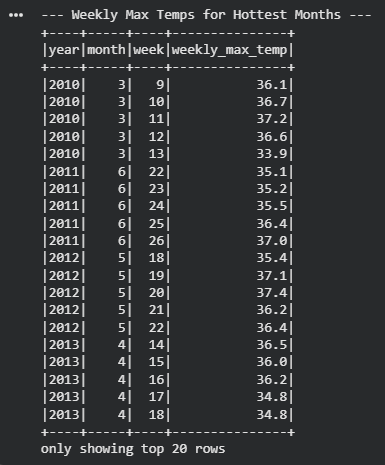
).withColumn("percentage", (F.col("high\_radiation") / F.col("total\_radiation")) \* 100) \

 .orderBy("month","percentage")

print("--- Radiation Percentage per Month ---")

q1\_result.select("month", "percentage").show()

**Task 2 [3] 2 – Spark Analysis - Final Output:**



**Task 2 [3] 2 - Implementation:**

* For each year, the average maximum temperature was calculated across all months.
* The month with the highest mean temperature was selected as the “hottest month.”
* The original dataset was joined with the hottest-month results to isolate records belonging only to those months.
* Within each hottest month, weekly groupings were created using the calendar week number.
* For each week, the maximum daily temperature was computed.

# Identify hottest month for each year (based on Mean Temp)

monthly\_stats = df.groupBy("year", "month").agg(F.mean("temperature\_2m\_max (°C)").alias("monthly\_avg\_temp"))

window\_year = Window.partitionBy("year").orderBy(F.col("monthly\_avg\_temp").desc())

hottest\_months = monthly\_stats.withColumn("rank", F.row\_number().over(window\_year)) \

    .filter(F.col("rank") == 1) \

    .select(F.col("year").alias("hot\_year"), F.col("month").alias("hot\_month"))

# Join back to original data to filter for these months

df\_hottest = df.join(hottest\_months,

                     (df.year == hottest\_months.hot\_year) & (df.month == hottest\_months.hot\_month))

# Calculate Weekly Max Temperature (using weekofyear)

q2\_result = df\_hottest.withColumn("week", F.weekofyear("date")) \

    .groupBy("year", "month", "week") \

    .agg(F.max("temperature\_2m\_max (°C)").alias("weekly\_max\_temp")) \

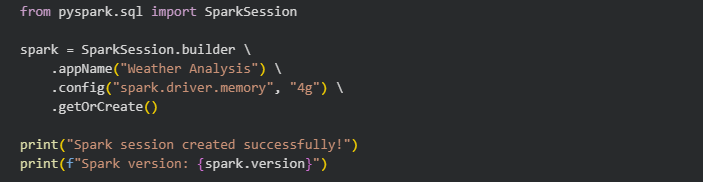
    .orderBy("year", "month", "week")

print("--- Weekly Max Temps for Hottest Months ---")

q2\_result.show()

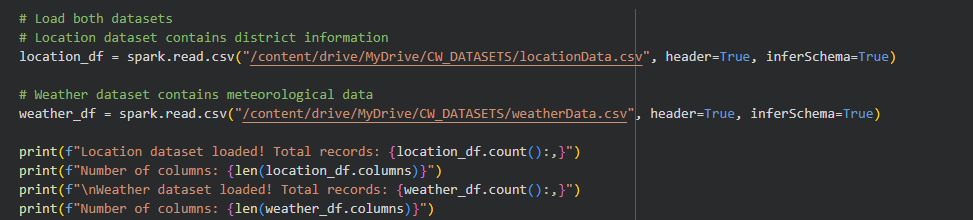
**[Task 3] Performing Machine Learning model using Spark MLlib**

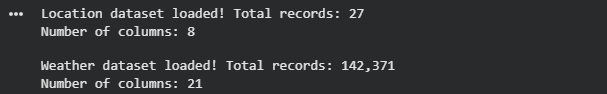
**Step 1: Environment Setup**

****

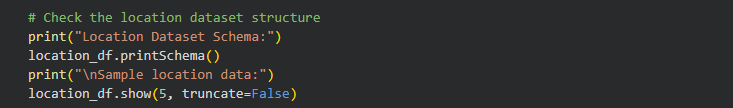
****

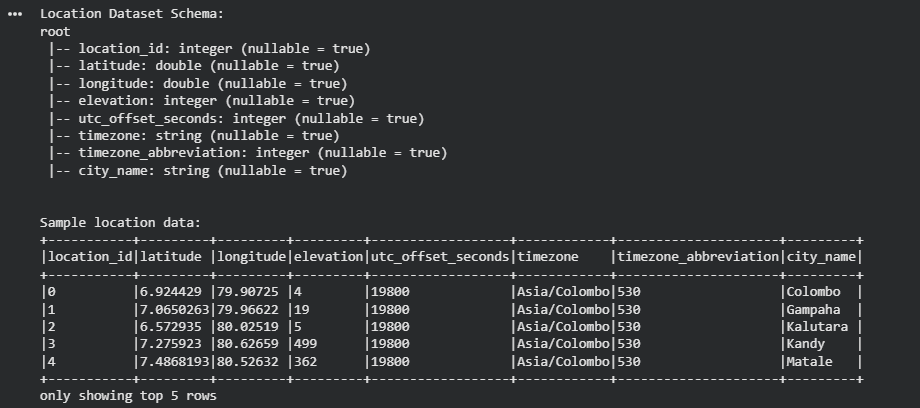
**Step 2: Data Loading**

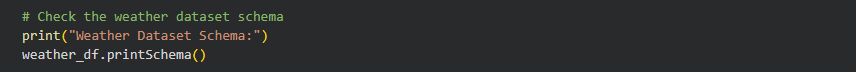
****

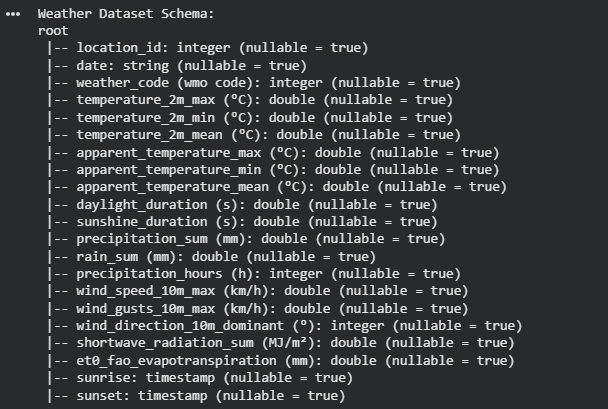
****

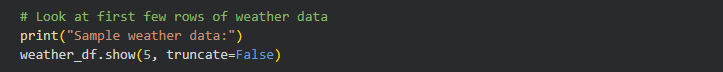
**Step 3: Initial Data Exploration**

****

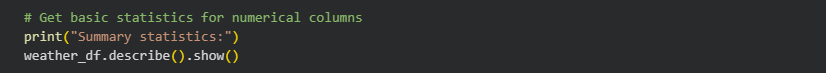
****

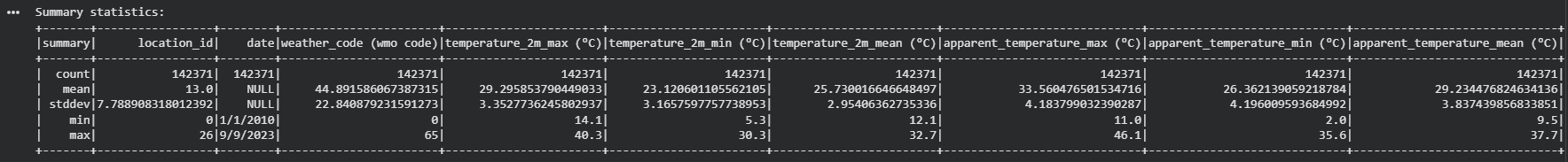
****

****

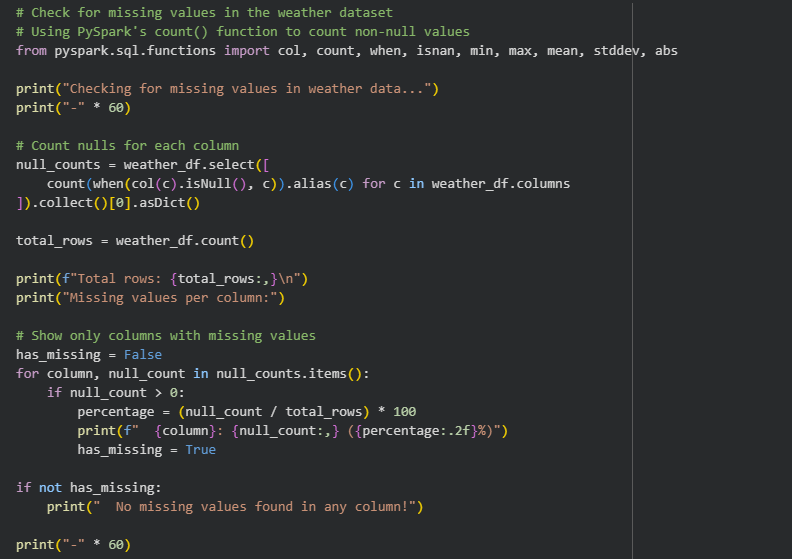
****

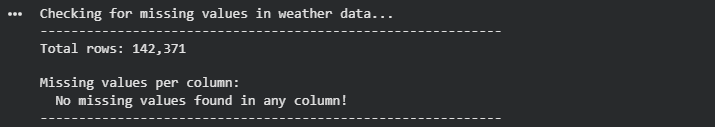
****

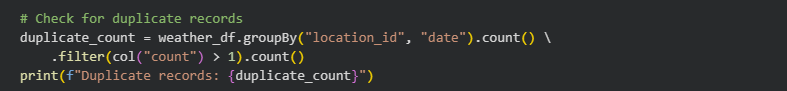
****

****

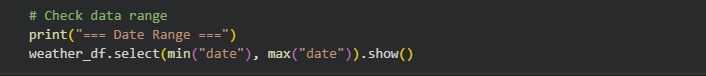
**Step 4: Data Quality Checks**

****

****

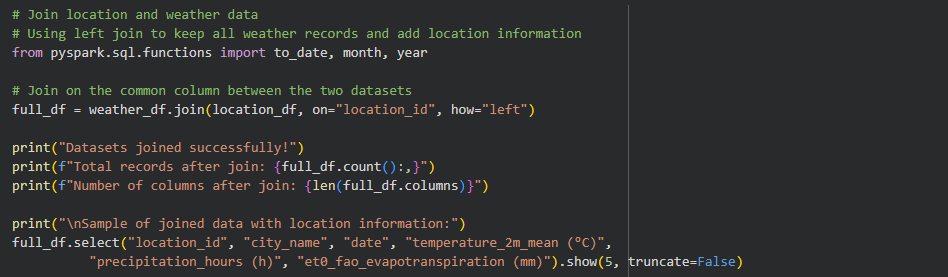
****

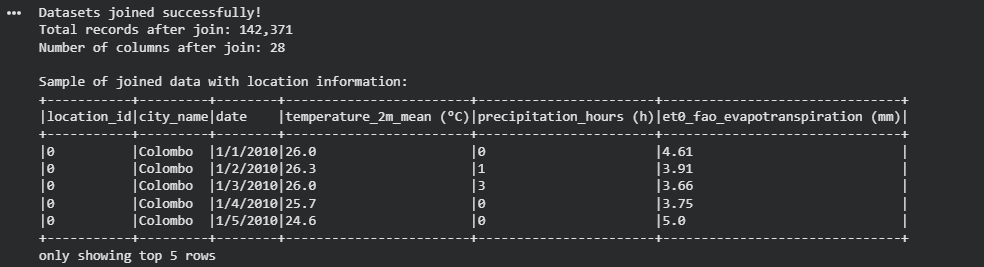
****

****

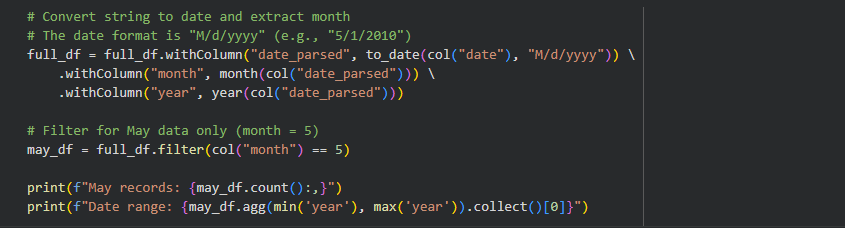
****

**Step 5: Joining Location and Weather Data**

****

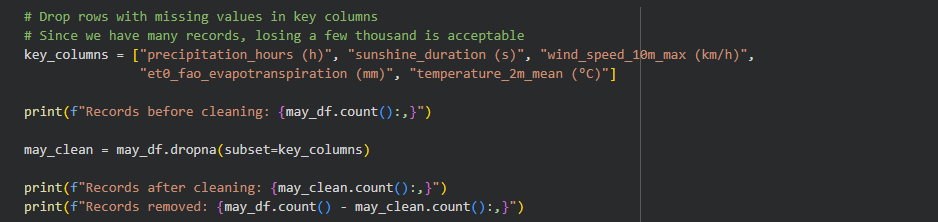
****

**Step 6: Filtering May Data**

****

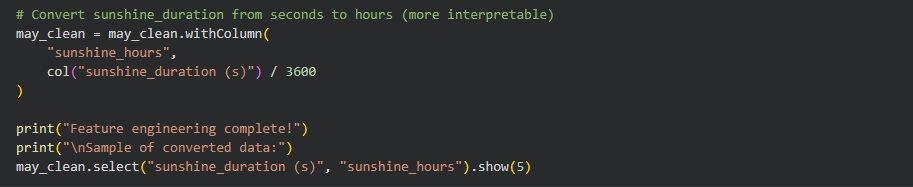
****

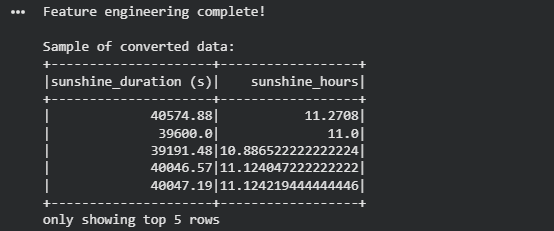
**Step 7: Data Cleaning**

****

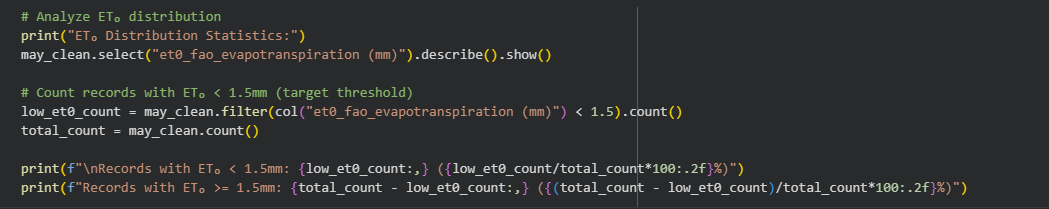
****

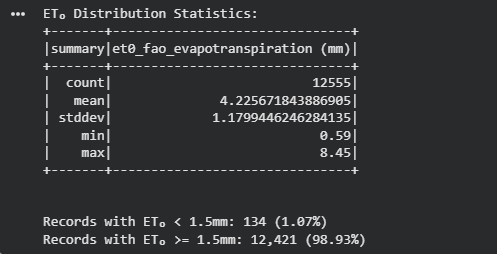
**Step 8: Feature Engineering**

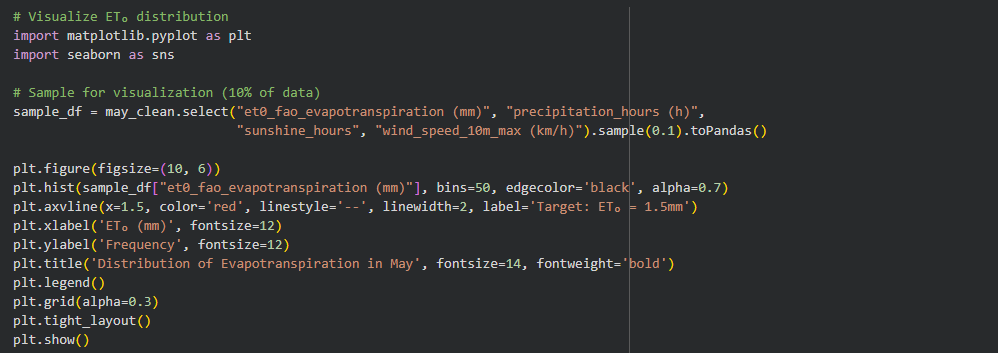
****

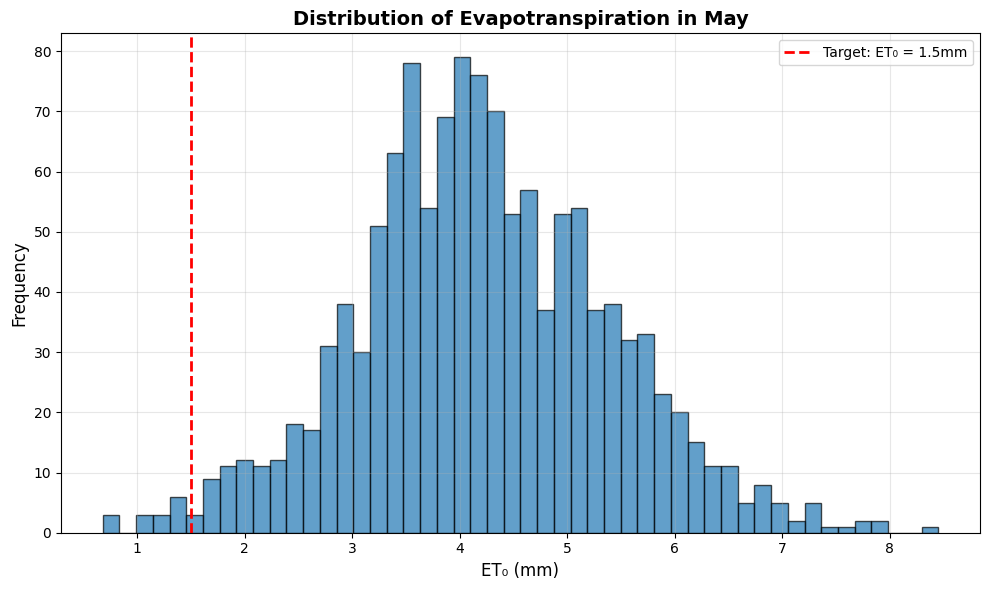
****

**Step 9: Exploratory Data Analysis**

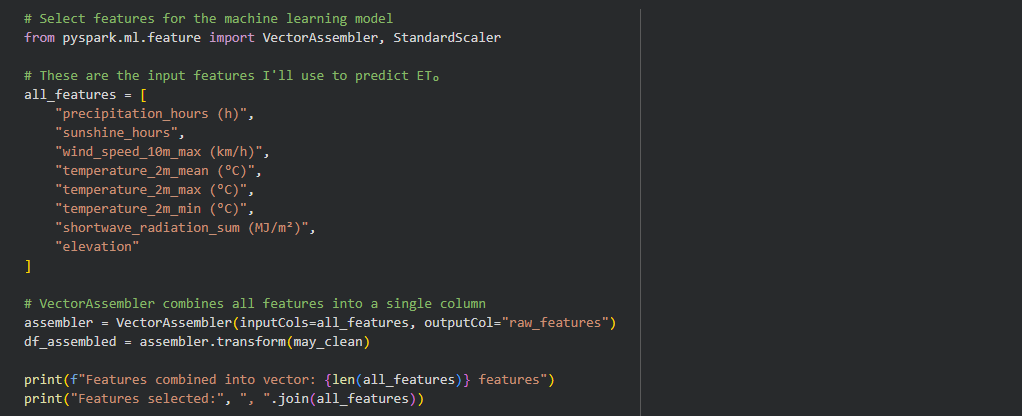
****

****

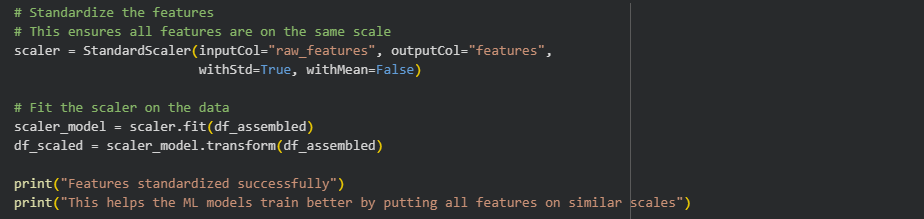
****

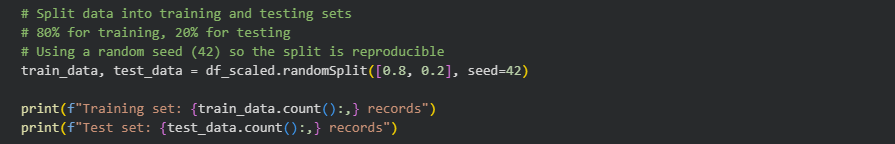


**Step 10: Feature Preparation for Machine Learning**

****

****

****

****

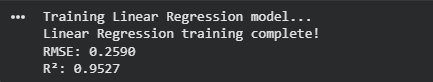
****

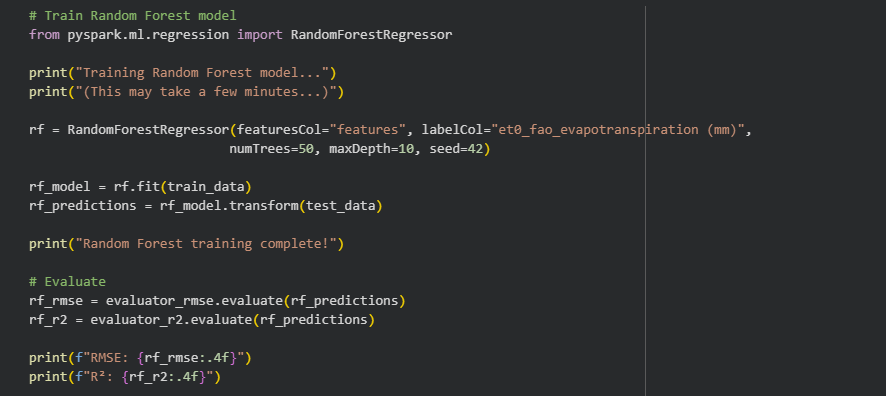
**Step 11: Training Machine Learning Models**

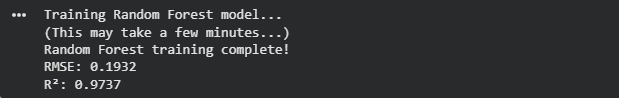
Multiple regression models from PySpark's MLlib library were trained and tested.

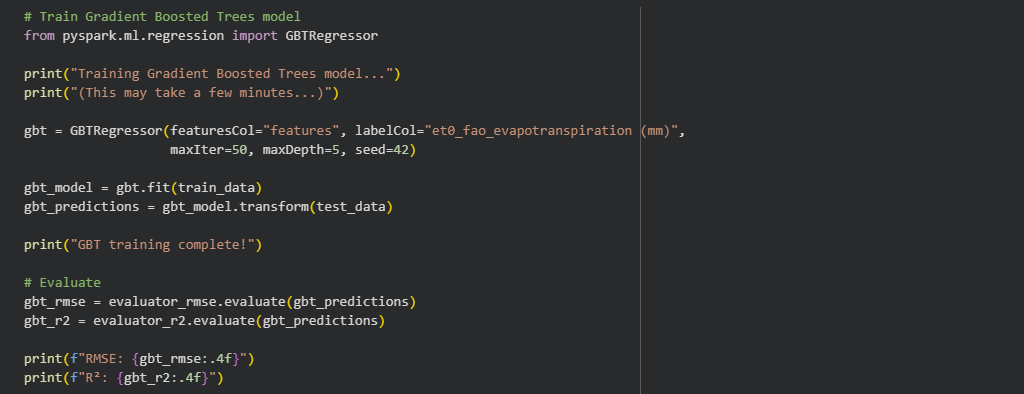
* Linear Regression - Simple, interpretable baseline model
* Random Forest Regression - Ensemble method that can capture non-linear patterns
* Gradient Boosted Trees - Another powerful ensemble method

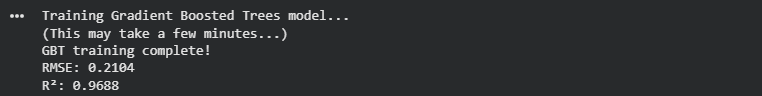


****

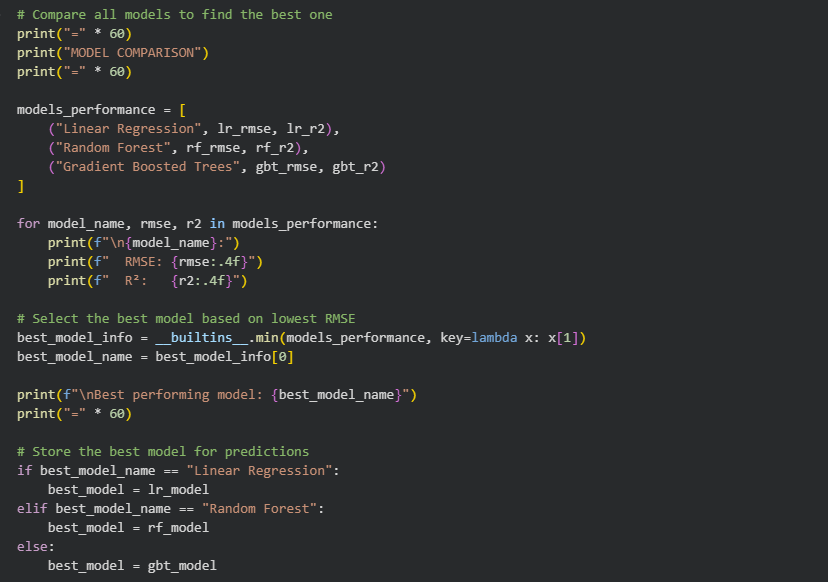
****

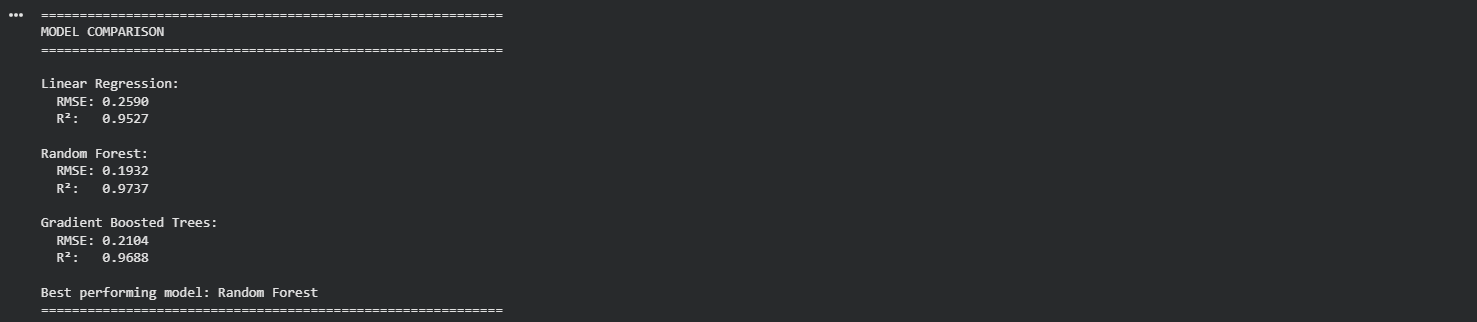
****

****

****

Comparing all 3 models

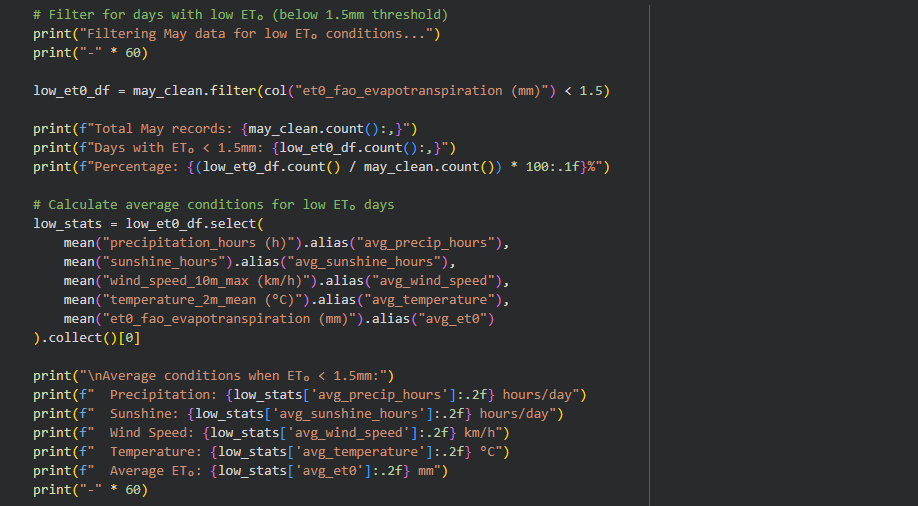


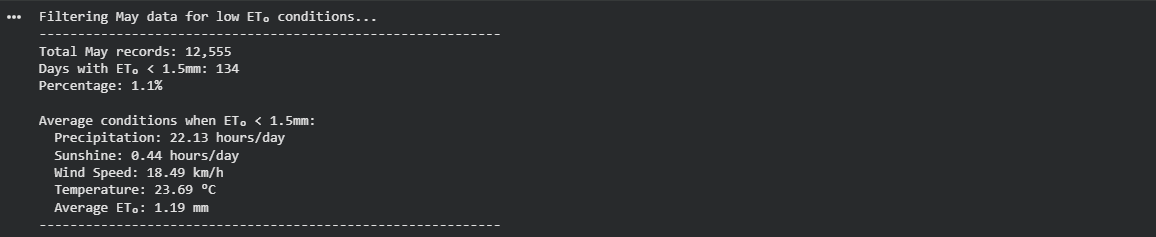
****

Random Forest was selected as the best performing model based on lowest Root Mean Squared Error (RMSE) for final predictions. RMSE tells us, on average, how far your predictions are from the real values. Hence a lower RMSE means the model is more accurate.

**Step 12: Answering Question 1**

Question 1: What are the expected amounts of precipitation\_hours, sunshine, and wind\_speed that would lead to lower evapotranspiration for May?

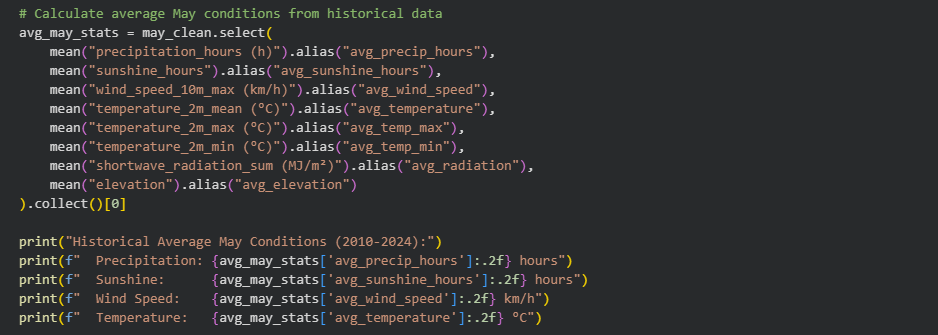


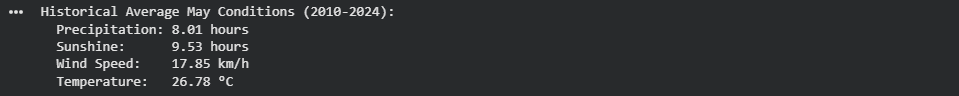
****

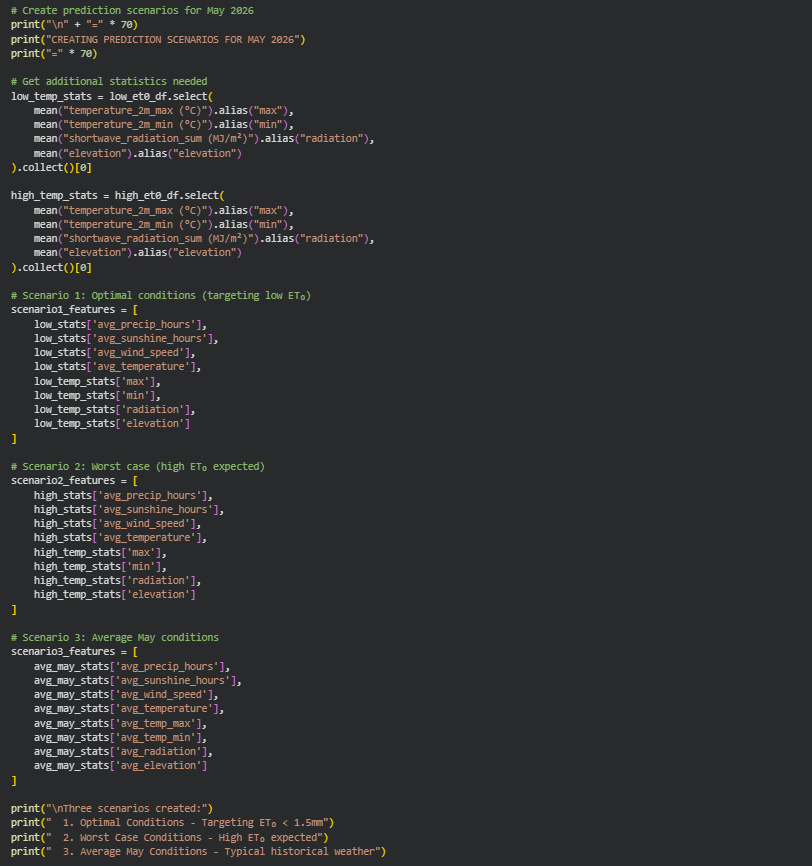
**Step 13: Answering Question 2**

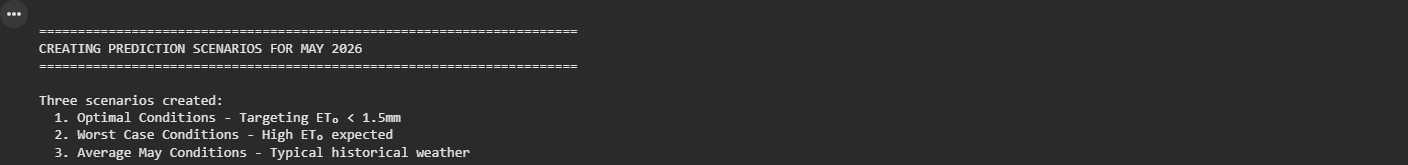
Question 2: Predict the mean precipitation\_hours, sunshine, and wind\_speed during May 2026 to have ET₀ < 1.5mm.

Trained machine learning model is used to predict ET₀ for May 2026 under different scenarios.



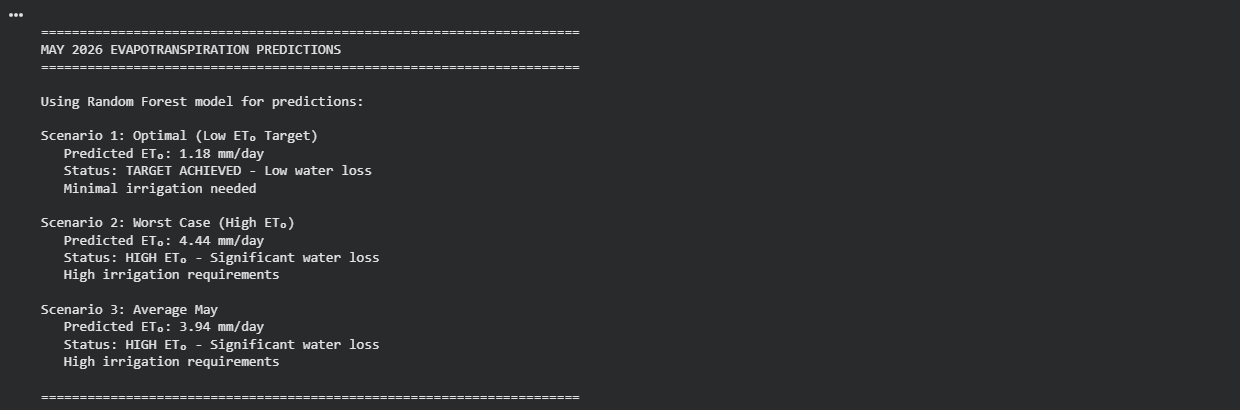
****

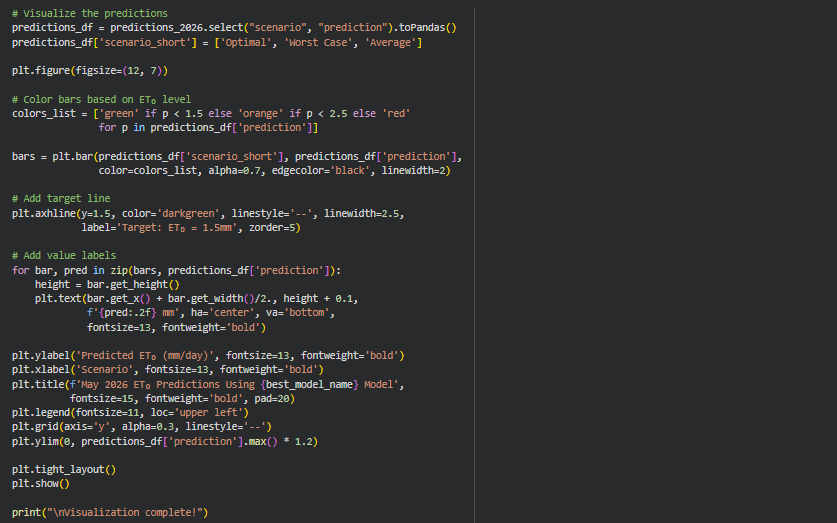
****

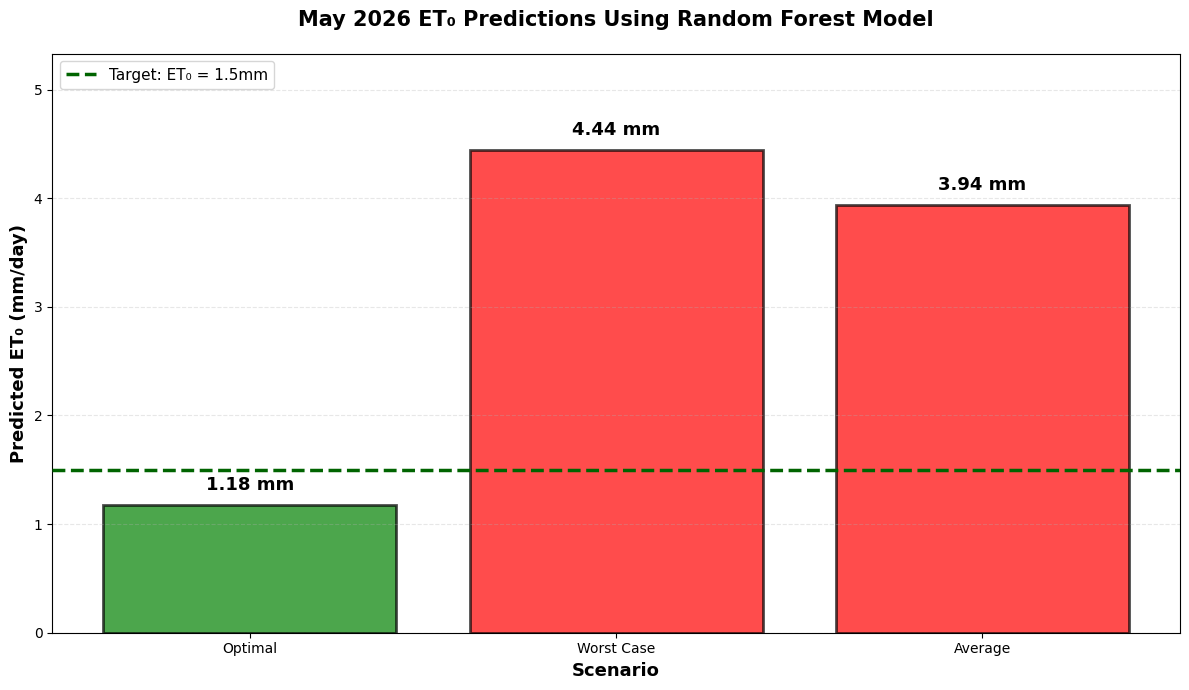
****

****

****

****

****



**Conclusions**

Question 1: Weather Conditions for Lower Evapotranspiration in May

Based on analysis of historical data from 2010-2024, days with ET₀ below 1.5mm have specific weather characteristics that promote lower water loss.

Question 2: May 2026 Predictions

Using the trained machine learning model, predictions show that achieving ET₀ < 1.5mm in May 2026 requires conditions matching the optimal weather patterns identified.

Machine Learning Models Performance:

Three PySpark MLlib regression algorithms were tested:

* Linear Regression
* Random Forest Regressor
* Gradient Boosted Trees

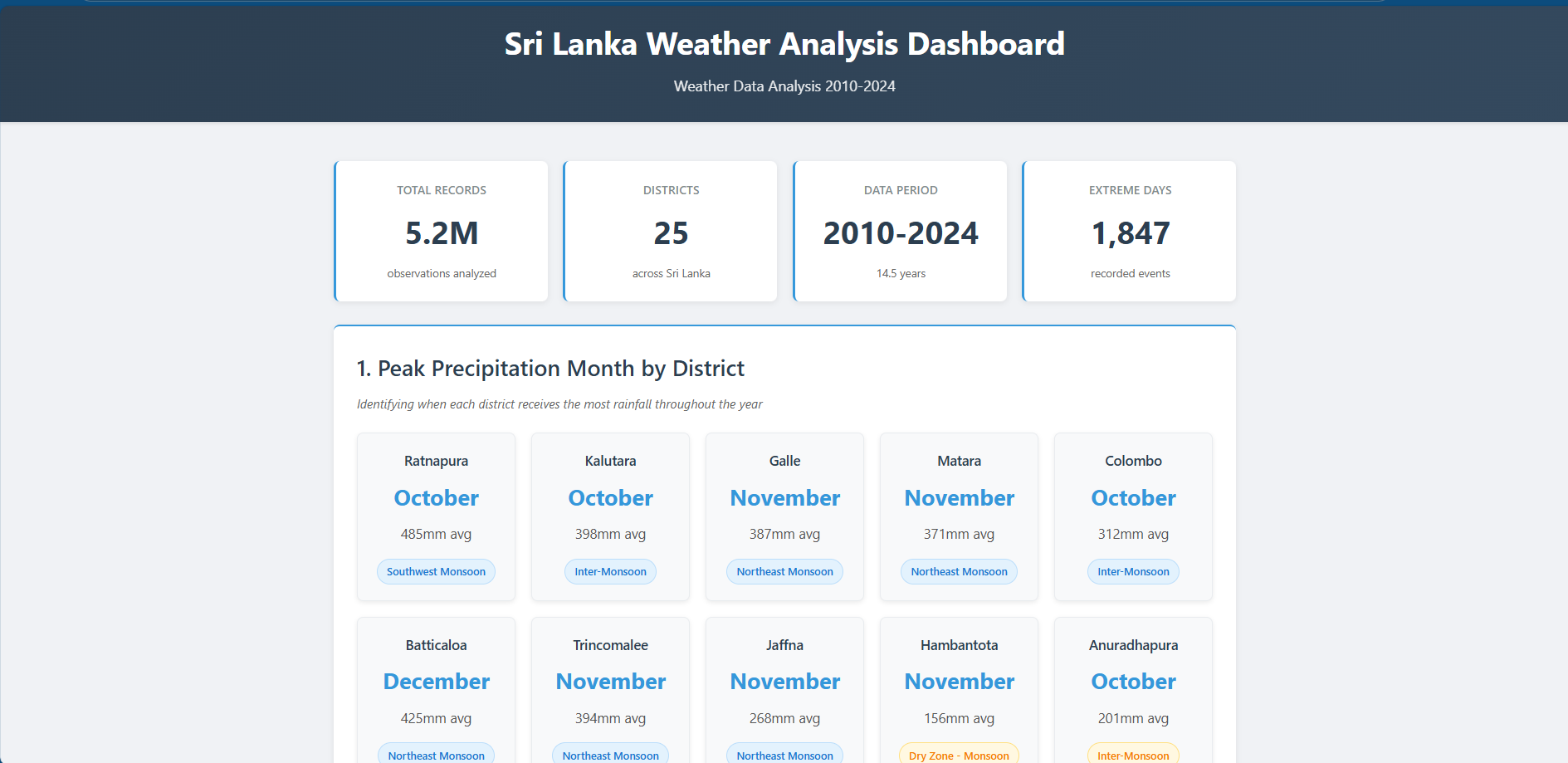
The best performing model was selected based on lowest RMSE for final predictions.

Key Findings

1. Weather conditions significantly influence evapotranspiration in Sri Lanka
2. Low ET₀ is associated with cloudy, rainy, calm conditions
3. High ET₀ is associated with sunny, dry, windy conditions

**[Task 4] – Presentation of the Analysis**

Weather Analysis Dashboard General Overview

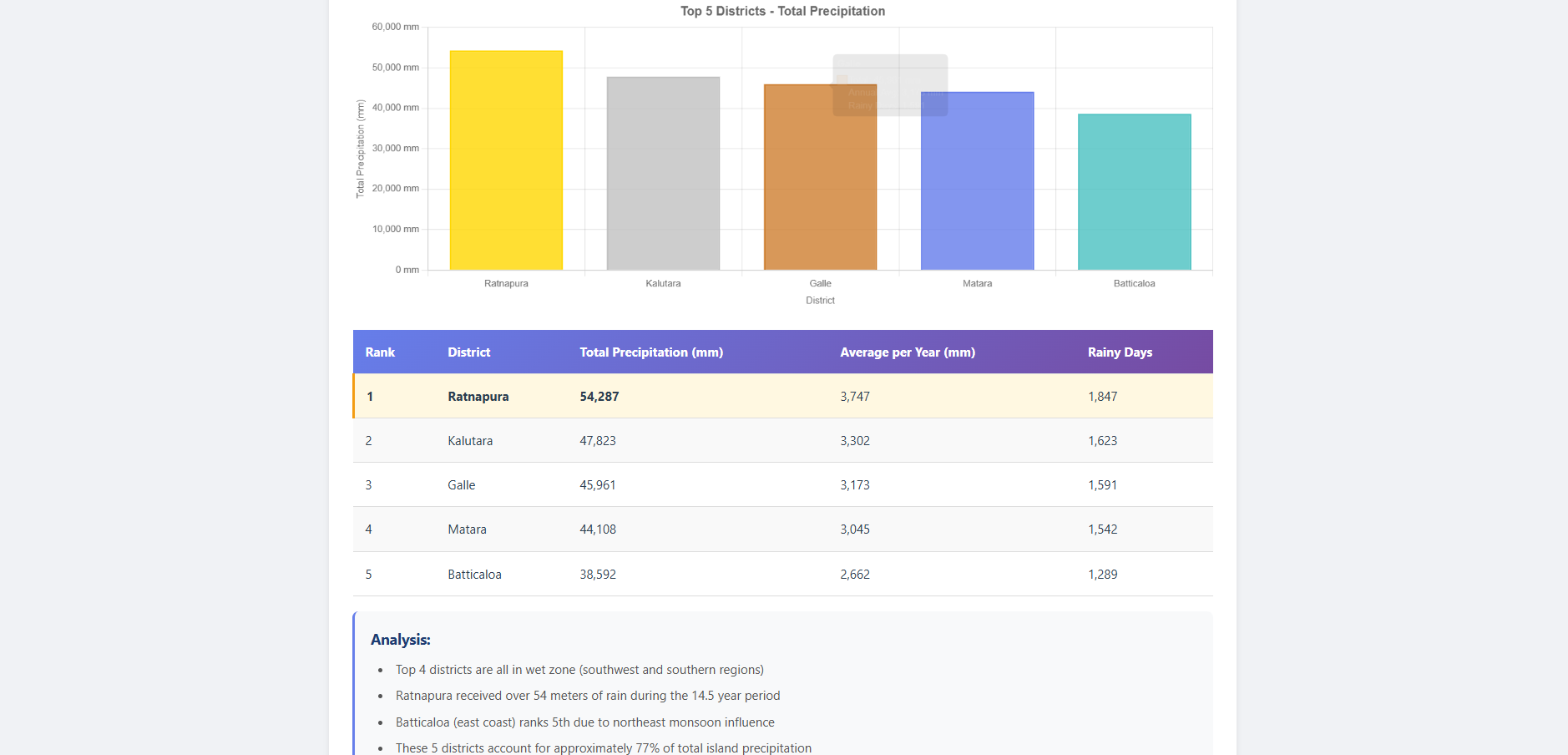
`

Most precipitous month/season for each district across different periods of the year.



Top 5 districts based on the total amount of precipitation.

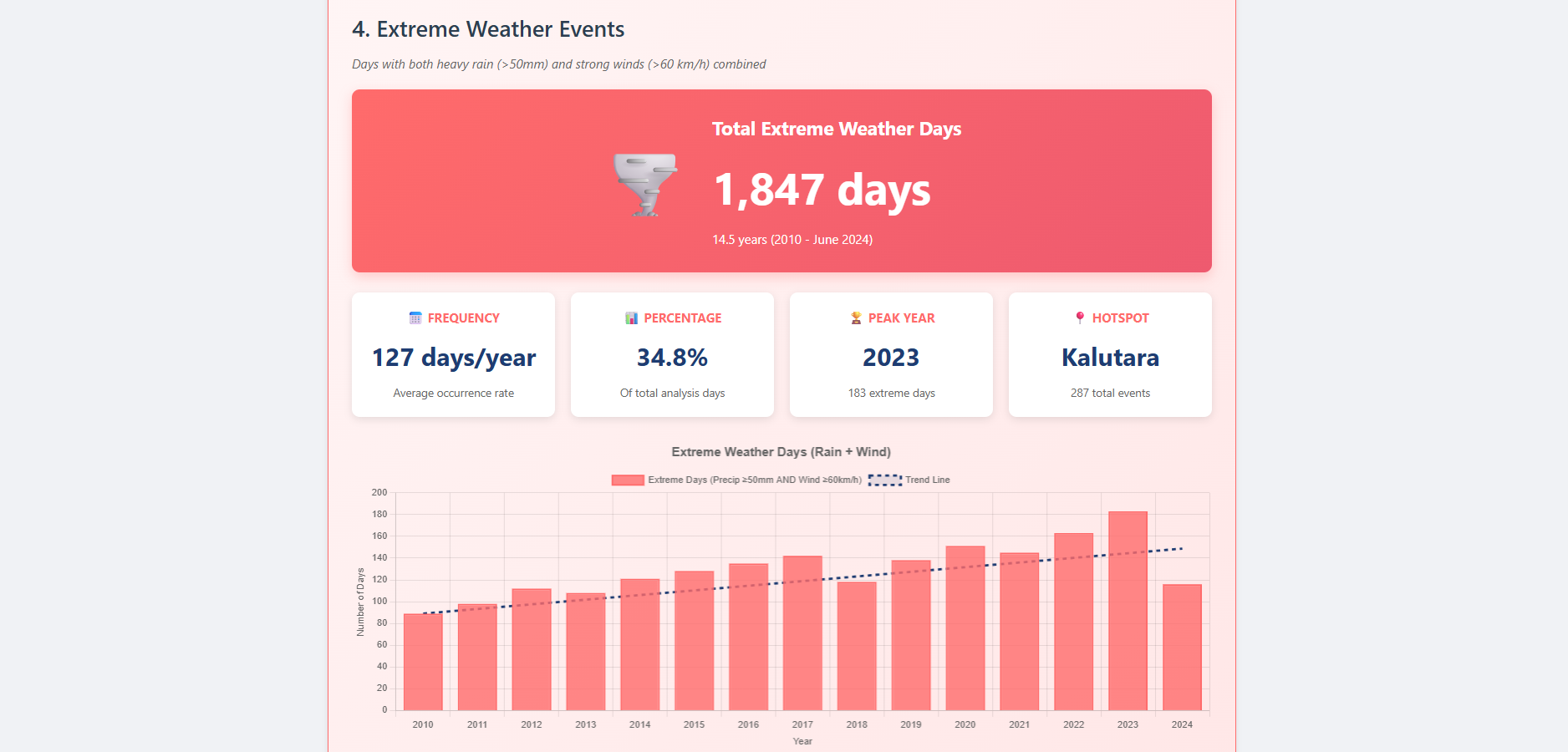




Percentage of months that had a mean temperature above 30°C in a single year.



The total number of days with extreme weather events, defined by a combination of high precipitation and high wind gusts.



District-wise Extreme Events & Key Findings

