📌 A. Kafka Producer (temperature1 topic)

producer.py

```
from time import sleep
from json import dumps
from kafka import KafkaProducer
import random
import datetime

producer = KafkaProducer(
   bootstrap_servers=['localhost:9092'],
   value_serializer=lambda x: dumps(x).encode('utf-8')
)

cities = ["Hyderabad","Bangalore","Chennai","Pune","Delhi","Mumbai","Kolkata",
        "Lucknow","Jaipur","Ahmedabad","Coimbatore","Mysore","Ranchi"]

while True:
   city = random.choice(cities)
   data = {
      "timestamp": str(datetime.datetime.now()),
      "city": city,
      "temperature": random.randint(20, 40),
      "humidity": random.randint(30, 85),
      "wind": random.randint(1, 25),
      "air_quality": random.randint(10, 150)
   }
```

```
    producer.send('temperature1', value=data)
    print("Sent:", data)
    sleep(2)
```

📌 B. Spark Structured Streaming Code (exact from your report)

spark_streaming.scala

```
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions._
import org.apache.spark.sql.types._

val spark = SparkSession.builder
    .appName("WeatherStreaming")
    .master("local[*]")
    .getOrCreate()

spark.sparkContext.setLogLevel("WARN")

// Read from Kafka
val df = spark.readStream
    .format("kafka")
    .option("kafka.bootstrap.servers", "localhost:9092")
    .option("subscribe", "temperature1")
    .load()

val schema = new StructType()
```

```
  .add("timestamp", StringType)
  .add("city", StringType)
  .add("temperature", IntegerType)
  .add("humidity", IntegerType)
  .add("wind", IntegerType)
  .add("air_quality", IntegerType)

val parsed = df.selectExpr("CAST(value AS STRING)")
  .select(from_json(col("value"), schema).as("data"))
  .select("data.*")

val avgTemp = parsed
  .groupBy("city")
  .agg(avg("temperature").alias("avg_temperature"))

// Write stream to console (for real-time insights)
val query = avgTemp.writeStream
  .outputMode("complete")
  .format("console")
  .start()

query.awaitTermination()
```

📌 C. Write Aggregated Temperature to CSV (same as PDF)

save_to_csv.scala

```
spark.sql("""
```

```
  SELECT * FROM TemperatureAvg
""")
.coalesce(1)
.write
.option("header", "true")
.mode("overwrite")
.csv("C:/Users/home/Desktop/TemperatureAvgCSV")
```

📌 D. Spark SQL Table Creation (from your report)

```
avgTemp.writeStream
  .outputMode("complete")
  .format("memory")
  .queryName("TemperatureAvg")
  .start()
```