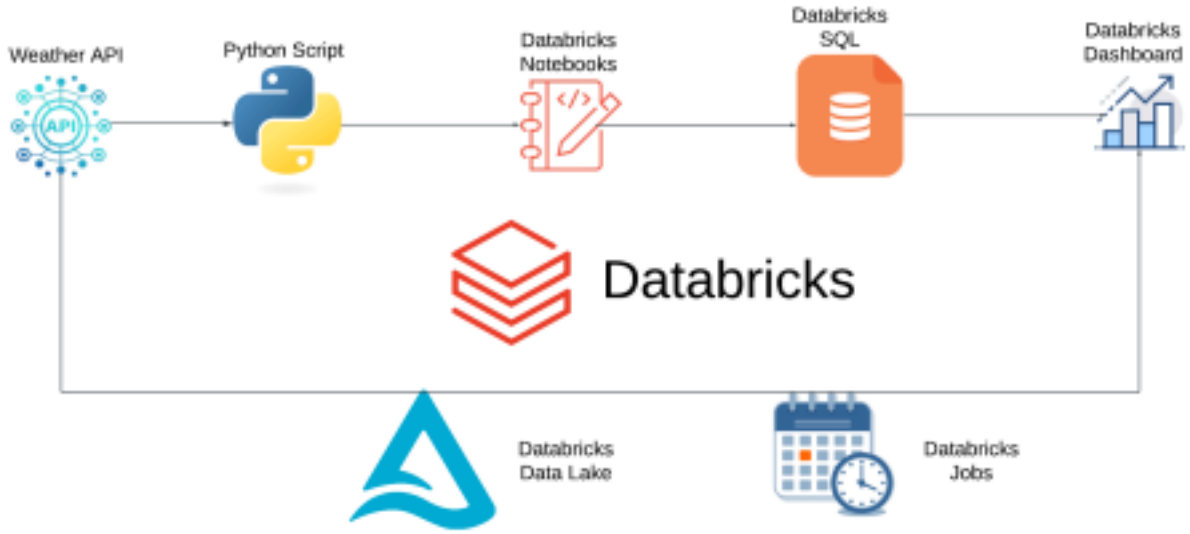
Workflow



Data Ingestion

```
# Install necessary libraries
import requests
import datetime
```

```
# API setup
API_KEY = 'c1d0f294ee92ad3c62e99d697f6f615e'
CITY = 'West Bengal'
URL = f"http://api.openweathermap.org/data/2.5/weather?q={CITY}&appid={API_KEY}"
```

Data Validation And Error Handling

```
# Function to fetch weather data
def fetch_weather_data():
    response = requests.get(URL)
    if response.status_code == 200:
        data = response.json()
        return {
            'city': data['name'],
            'temperature': data['main']['temp'],
            'description': data['weather'][0]['description'],
            'timestamp': datetime.datetime.now().isoformat()
        }
    else:
        print(f"Error: {response.status_code}")
        return None
```

Data Storage

Data Preprocessing and Cleaning

Data Analysis and Aggregation

```
%scala
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions._
val weatherDataPath = "/mnt/delta/weather_data"
val rawData = spark.read.format("delta").load(weatherDataPath)
rawData.show()
val cleanedData = rawData
  .select(
    col("city"),
    col("temperature").cast("double"),
    col("description"),
    col("timestamp").cast("timestamp")
  )
  .na.drop()  // Drop rows with any missing values

// Calculate the average temperature per city
val averageTemp = cleanedData.groupBy("city").agg(avg("temperature").alias("avg_temperature"))

// Display the result
averageTemp.show()
```

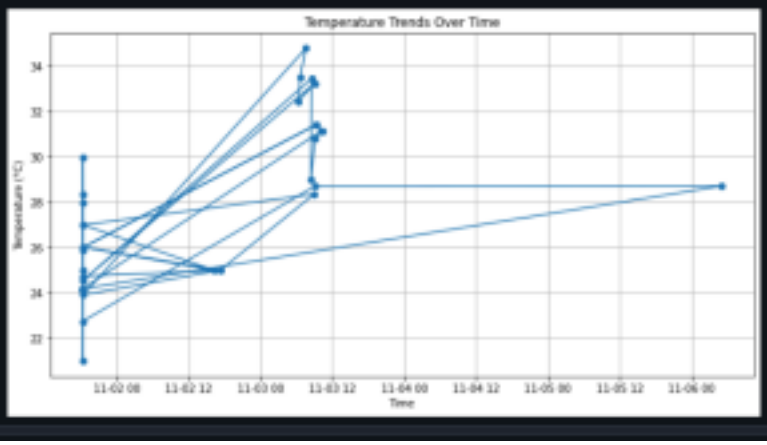
```
(3) Spark Jobs
rawData: org.apache.spark.sql.DataFrame = [city: string, description: string ... 2 more fields]
cleanedData: org.apache.spark.sql.DataFrame = [city: string, temperature: double ... 2 more fields]
averageTemp: org.apache.spark.sql.DataFrame = [city: string, avg_temperature: double]
+-----------+---------------+-----------+-------------------+
|       city|    description|temperature|          timestamp|
+-----------+---------------+-----------+-------------------+
```

Data Visualization

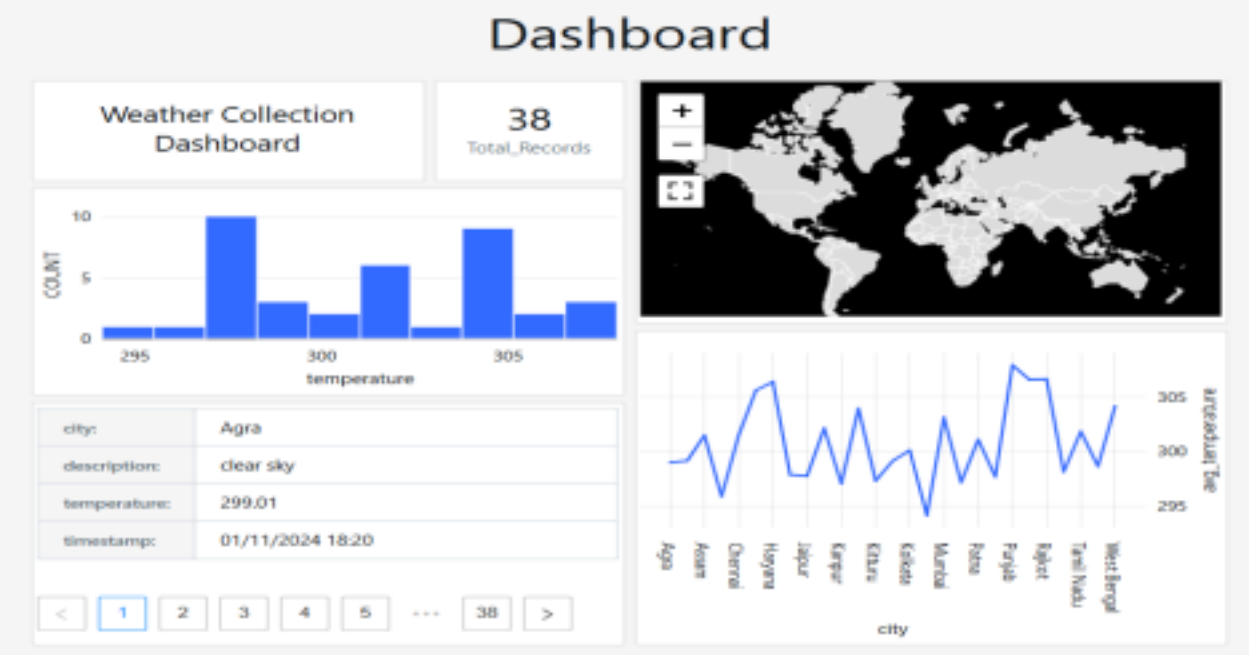
```
df['temperature_celsius'] = df['temperature'] - 273.15

# Plot temperature trends over time
plt.figure(figsize=(12, 6))
plt.plot(pd.to_datetime(df['timestamp']), df['temperature_celsius'], marker='o', linestyle='-')
plt.title('Temperature Trends Over Time')
plt.xlabel('Time')
plt.ylabel('Temperature (°C)')
plt.grid(True)
plt.show()
```

```
(1) Spark Jobs
```



Building Report Using Dashboard



Monitoring Pipeline

Automate Collection