**2 .Real-Time Data Processing System for Weather Monitoring**:

import requests

import time

import sqlite3

import pandas as pd

import smtplib

import matplotlib.pyplot as plt

from datetime import datetime

# Initialize SQLite database connection

conn = sqlite3.connect(&quot;weather\_data.db&quot;)

cursor = conn.cursor()

# Create table if not exists

cursor.execute(&#39;&#39;&#39;CREATE TABLE IF NOT EXISTS weather\_data (

city TEXT,

date DATE,

main TEXT,

temp REAL,

feels\_like REAL,

max\_temp REAL,

min\_temp REAL,

rain\_percentage REAL

)&#39;&#39;&#39;)

# Configurations

API\_KEY = &quot;dbc2dfa8b7ebd6327c4203d3e15f9362&quot;

ALERT\_THRESHOLD = 35.0 # Example threshold

# Define a function to retrieve weather forecast data by city name

def get\_weather\_forecast\_by\_city(city\_name):

url = f&quot;http://api.openweathermap.org/data/2.5/forecast?q={city\_name}&amp;appid={API\_KEY}&quot;

response = requests.get(url)

if response.status\_code == 200:

data = response.json()

return data

else:

print(f&quot;Failed to retrieve forecast data for {city\_name}. Status Code: {response.status\_code}, Response: {response.text}&quot;)

return None

# Analyze the weather forecast for the specified day

def analyze\_weather\_forecast(forecast\_data, user\_date):

weather\_summary = {}

for item in forecast\_data[&#39;list&#39;]:

timestamp = item[&#39;dt&#39;]

main\_condition = item[&#39;weather&#39;][0][&#39;main&#39;]

temp = item[&#39;main&#39;][&#39;temp&#39;] - 273.15

feels\_like = item[&#39;main&#39;][&#39;feels\_like&#39;] - 273.15

rain = item.get(&#39;rain&#39;, {}).get(&#39;1h&#39;, 0)

# Collect weather data

date = time.strftime(&#39;%Y-%m-%d&#39;, time.gmtime(timestamp))

if date not in weather\_summary:

weather\_summary[date] = {

&quot;conditions&quot;: [],

&quot;temp&quot;: 0,

&quot;feels\_like&quot;: 0,

&quot;rain&quot;: 0,

&quot;count&quot;: 0 # Count how many forecasts there are for the day

}

weather\_summary[date][&quot;conditions&quot;].append(main\_condition)

weather\_summary[date][&quot;temp&quot;] += temp

weather\_summary[date][&quot;feels\_like&quot;] += feels\_like

weather\_summary[date][&quot;rain&quot;] += rain

weather\_summary[date][&quot;count&quot;] += 1 # Increase count

# Filter for the specific user-provided date

return weather\_summary.get(user\_date, None)

# Continuous loop for user input

while True:

user\_city = input(&quot;Enter the city name to check the weather forecast (or type &#39;exit&#39; to quit): &quot;).strip()

if user\_city.lower() == &quot;exit&quot;: # Check for exit condition

print(&quot;Exiting the program.&quot;)

break

user\_date = input(&quot;Enter the date in YYYY-MM-DD format (or press Enter for today&#39;s date): &quot;).strip()

if user\_date == &quot;&quot;:

user\_date = time.strftime(&#39;%Y-%m-%d&#39;) # Default to today&#39;s date

# Fetch weather forecast data for the specified city

forecast\_data = get\_weather\_forecast\_by\_city(user\_city)

if forecast\_data:

weather\_summary = analyze\_weather\_forecast(forecast\_data, user\_date)

# Display the summary of the day&#39;s weather

print(f&quot;\nWeather Forecast Summary for {user\_city} on {user\_date}:&quot;)

print(&quot;---------------------------------------&quot;)

if weather\_summary:

avg\_temp = weather\_summary[&quot;temp&quot;] / weather\_summary[&quot;count&quot;]

avg\_feels\_like = weather\_summary[&quot;feels\_like&quot;] / weather\_summary[&quot;count&quot;]

rain\_amount = weather\_summary[&quot;rain&quot;] # Total rain for the day

# Determine the most frequent weather condition

most\_common\_condition = max(set(weather\_summary[&quot;conditions&quot;]), key=weather\_summary[&quot;conditions&quot;].count)

# Print data in sequence

print(f&quot;{user\_date}:&quot;)

print(f&quot; Main Condition: {most\_common\_condition}&quot;)

print(f&quot; Avg Temp: {avg\_temp:.1f}°C&quot;)

print(f&quot; Avg Feels Like: {avg\_feels\_like:.1f}°C&quot;)

print(f&quot; Total Rain: {rain\_amount:.1f}mm&quot;)

else:

print(f&quot;No data available for {user\_date}.&quot;)

print(&quot;---------------------------------------\n&quot;)

# Close the database connection after exiting the loop

conn.close()