```
const express = require('express');
const axios = require('axios');
const app = express();
const port = 3000;
const key = 'ba8c466e3b9549c044e27d3cbe3f04b2';

app.use(express.json());
app.use(express.static('public'))

function formatingData(dateDict) {
    let dataToReturn = [], averageTemps = [];
    let umbrella = "No", mask = "No";

    for(date in dateDict) {
        let data = [date];
        let mainWeather = dateDict[date].shift();
        let pollution = dateDict[date].shift();

        if(mainWeather.includes('Rain')) { umbrella = "Yes"; }

        let temps = dateDict[date].map( (v) => v.slice(0, 1) );
        temps = [].concat.apply([], temps);
        let windSpeed = dateDict[date].map( (v) => v.slice(1, -1) );
        windSpeed = [].concat.apply([], windSpeed);
        let rainfall = dateDict[date].map( (v) => v.splice(2, 1) );
        rainfall = [].concat.apply([], rainfall);

        let avgTemp = temps.reduce((a, b) => a + b, 0) / temps.length;
        averageTemps.push(avgTemp);

        if( Math.max(pollution) >= 10.0 ) { mask = "Yes" }

        data.push(Math.min(...temps));
        data.push(Math.max(...temps));
        data.push(Math.max(...windSpeed));
        data.push(rainfall.reduce((a, b) => {
            if(b != undefined) { return a + b } else {return a }
        }, 0));

        dataToReturn.push(data);
    }
    dataToReturn.unshift(mask);
    let avgTemps = averageTemps.reduce((a, b) => a + b, 0) / averageTemps.length;
    if(avgTemps < 10.0) { dataToReturn.unshift("Cold") }
    else if(avgTemps >= 10.0 && avgTemps <= 20.0) { dataToReturn.unshift("Warm") }
    else dataToReturn.unshift("Hot");
    dataToReturn.unshift(umbrella);

    return dataToReturn;
}

app.get('/', function(req, res) {
    res.sendFile(__dirname + "/" + "index.html");
})

app.put('/city', function(req, res) {
    let city = req.body.city, data, lat, long;

    axios({
        method: 'get',
        url: 'https://api.openweathermap.org/data/2.5/forecast',
        params: {
            q: city,
            appid: key,
            units: 'metric',
        }
    }).then(function(response) {
        data = response.data;

        // just going to group the data into a dict, where the key is the date
that some forecast belongs too, the value will be a list containing all data we
are concerned with for that date
        dataPoints = data.list;
        let dateDict = {};

        for(d in dataPoints) {
            let dta = dataPoints[d];
            let date = dta.dt_txt.substr(5, 5).split('');
            [date[0], date[1], date[2], date[3], date[4]] = [date[3], date[4],
date[2], date[0], date[1]];
            date = date.join('');
            if( !(date in dateDict) ) {
                dateDict[date] = [];
                // adding two empty lists that will contain the main weather and
the pollution data
                dateDict[date].push([]);
                dateDict[date].push([]);
            }

            dateDict[date][0].push(dta.weather[0].main);

            let rain = dta.rain;
            // if there is no rain during the 3 hour period, there will be no
rain field, otherwise there will be a rain field and a 3h key within it that will
contain the data
            if(rain != undefined) {
                dateDict[date].push([dta.main.temp, dta.wind.speed, rain['3h']])
            } else {
                dateDict[date].push([dta.main.temp, dta.wind.speed, rain])
            }
        }
        lat = data.city.coord.lat, long = data.city.coord.lon
        axios({
            method: 'get',
            url: `https://api.openweathermap.org/data/2.5/air_pollution/forecast?
lat=${lat}&lon=${long}&appid=${key}`,
        }).then(function(response) {
            dataPoints = response.data.list;
            // put the pollution data into the correct date in the dateDict
            for(d in dataPoints) {
                let dta = dataPoints[d];
                let date = new Date(null);
                date.setSeconds(dta.dt);
                date = date.toISOString().substr(5,5).split('');
                [date[0], date[1], date[2], date[3], date[4]] = [date[3],
date[4], date[2], date[0], date[1]];
                date = date.join('');
                if(dateDict[date] != undefined) {
                    dateDict[date][1].push(dta.components.pm2_5);
                }
            }
            data = formatingData(dateDict);
            res.send(data);
        }).catch(function(error) {
            console.log(error);
            res.send(data);
        })
    }).catch(function(error) {
        console.log(error)
        res.end()
    })
})

app.listen(port, () => console.log(`Example app listening on port ${port}!`));
```