**CODE EXPLANATION**

**Part 1: Data Collection, Pre-processing and Handling (30 marks):**

**From .txt to cleaned csv files**  
This code snippet performs a series of operations on CSV files stored within a directory. Here's a breakdown of what each part does:

1. **Function to Convert Text Files to CSV (convert\_txt\_to\_csv):**
   * This function is defined to convert text files in a specified input folder into CSV format and save them to an output folder.
   * It utilizes the Pandas library (**pd**) to read fixed-width format text files (**pd.read\_fwf**) and save them as CSV files (**df.to\_csv**).
   * The function takes two arguments: **input\_folder**, the directory containing the text files, and **output\_folder**, the directory where the resulting CSV files will be saved.
   * Before processing, it ensures that the output folder exists. If not, it creates the folder using **os.makedirs**.
2. **Usage of convert\_txt\_to\_csv Function:**
   * Here, the function is invoked with specific input and output folder paths.
   * **input\_folder** is set to **'DataAE1\_1March\_deadline'**, presumably containing text files to be converted.
   * **output\_folder** is set to **'DataAE1.csv'**, which seems to be the desired location to save the resulting CSV files.
3. **Cleaning CSV Files:**
   * After the text-to-CSV conversion, another process is initiated to clean CSV files.
   * Input directory (**input\_directory**) is set to **'DataAE1.csv'**, which likely contains the CSV files generated earlier.
   * Output directory (**output\_directory**) is set to **'DataAE1csv.cleaned'**, where cleaned CSV files will be stored.
   * The script iterates through each file in the input directory, identified by the **.csv** extension.
   * For each CSV file, it reads the file into a Pandas DataFrame (**pd.read\_csv**).
   * It then replaces any occurrences of **'---'** with **0** in the DataFrame using **df.replace('---', 0, inplace=True)**.
   * The modified DataFrame is saved to a new CSV file in the output directory, with the prefix **'cleaned\_'**.

Overall, this code performs data conversion and cleaning tasks on CSV files, leveraging the capabilities of the Pandas library for data manipulation and handling file operations using **os** module for directory management.

**Part 2: Average Temperature Change (15 marks)**

The provided Python script analyzes temperature data from multiple weather stations, following these steps:

1. **Data Loading:** The script navigates through a directory containing CSV files, each representing a weather station's temperature records. It loads each CSV file into a pandas DataFrame, converting temperature columns ('tmax' and 'tmin') to numeric values.
2. **Temperature Change Calculation:** For each station, the script calculates the average temperature change by comparing the initial and final temperatures recorded in the dataset. It computes the percentage change for both maximum and minimum temperatures and averages them to obtain the overall temperature change percentage.
3. **Identifying Extremes:** The script identifies the station with the greatest and lowest average temperature changes by finding the maximum and minimum values among the calculated temperature change percentages.
4. **Visualization:** To visualize temperature trends over time for each station, the script creates a grid of subplots. Each subplot represents a station and plots both maximum and minimum temperatures over the years. This allows for a clear comparison of temperature variations among stations without overcrowding the plot with a large legend.
5. **Plot Optimization:** To accommodate varying numbers of stations, the script dynamically adjusts the layout of subplots based on the number of stations. It ensures that empty subplots are hidden, providing a clean and concise visualization.

Overall, the script efficiently processes temperature data from multiple stations, calculates average temperature changes, identifies extremes, and presents temperature trends visually, facilitating comprehensive analysis and interpretation of temperature variations across different geographical locations.

**Part 3:** **Station Rankings (15 marks):**

This Python script analyzes temperature data from multiple weather stations, focusing on the top-k stations with the highest maximum temperatures:

1. **Data Loading:** The script reads temperature data from CSV files stored in a directory, loading each file into a pandas DataFrame. It converts temperature values to numeric format for analysis.
2. **Ranking Stations:** The script identifies the stations with the highest maximum temperatures by extracting the maximum temperature recorded for each station. It ranks the stations in descending order based on their maximum temperatures.
3. **Top-k Selection:** It selects the top-k stations with the highest maximum temperatures for further analysis. The value of **top\_k** can be adjusted to specify the number of top stations to consider.
4. **Temperature Distribution Plot:** The script creates a histogram to visualize the distribution of maximum temperatures for the selected top-k stations. Each station's temperature data is plotted as a separate histogram, allowing comparison of temperature distributions among stations. The histogram's x-axis represents maximum temperature values, and the y-axis shows the density of temperature occurrences.
5. **Summary Statistics:** For each selected station, the script calculates summary statistics including mean, median, and standard deviation of the maximum temperatures. These statistics provide insights into the central tendency and variance of temperature data for each station.

Overall, the script provides a comprehensive analysis of temperature distribution among the top-k stations with the highest maximum temperatures, enabling insights into temperature trends and variations across different geographical locations. Adjustments to **top\_k** can be made to focus on a specific number of top stations for analysis.

**Part 4**

The provided Python code conducts a structured analysis of temperature data from various weather stations, aiming to uncover anomalies or unexpected patterns that might offer insights into climate trends. Here's a detailed breakdown of the code's functionality:

**1. Data Loading:** The script begins by accessing a directory containing CSV files, each representing temperature records from different weather stations. It iterates through these files, loading the data into pandas DataFrames, and ensuring that temperature columns ('tmax' and 'tmin') are treated as numeric values. This step organizes the data into a dictionary, with each station's name serving as a key and its associated DataFrame as the corresponding value.

**2. Identifying Anomalies:** Once the data is loaded, the script proceeds to identify anomalies, defined as instances where temperature observations deviate significantly from the expected norms. To accomplish this, the script calculates z-scores for maximum temperatures ('tmax'). These z-scores represent the number of standard deviations each temperature observation is from the mean. Years with z-scores exceeding a predetermined threshold (set at 2 in this case) are flagged as having unusually high temperatures, potentially indicating anomalies.

**3. Discussion of Limitations:** Following the anomaly detection process, the script engages in a thoughtful discussion regarding the limitations associated with interpreting these anomalies in the context of climate change. It iterates through each station, examining the anomaly years identified earlier, and presents a series of considerations:

* **Potential Causes:** Anomalies could arise from various factors, including data errors, equipment malfunctions, or short-term weather phenomena. It's essential to account for these possibilities before attributing anomalies to broader climate trends.
* **Isolated Nature:** Anomalies observed in specific years or stations may not necessarily signify long-term climate shifts. They could be isolated occurrences without significant implications for overall climate patterns.
* **Data Quality Concerns:** Reliable climate analysis hinges on high-quality data. Factors like urbanization, station relocation, and data inconsistencies can introduce biases, necessitating careful consideration when interpreting anomalies.
* **Nature of Climate Change:** Climate change impacts typically manifest gradually over time, influencing trends rather than isolated observations. While anomalies may offer valuable insights, they should be interpreted within the broader context of climate change dynamics.
* **Call for Further Investigation:** Anomalies should serve as prompts for further investigation rather than definitive evidence of climate change. Rigorous research and analysis are required to understand the underlying causes and implications of anomalies fully.

**Conclusion:** In summary, the script employs a systematic approach to identify anomalies in temperature data and engages in a nuanced discussion regarding their interpretation in the context of climate change. By considering various factors and emphasizing the need for cautious analysis, the script contributes to a more informed understanding of climate dynamics and informs evidence-based decision-making in climate science and policy.

**Part 5:**

Top of Form

Bottom of Form