To simulate 100,000 lines of IoT sensor data and provide analysis questions, I'll generate data for a hypothetical smart home scenario. The sensors will include:

1. **Temperature Sensor:** Degrees Celsius, realistic range.
2. **Humidity Sensor:** Percentage, realistic range.
3. **Light Sensor:** Lux, representing ambient light levels.
4. **Motion Sensor:** Binary (0/1), indicating presence.
5. **Door/Window Sensor:** Binary (0/1), indicating open/closed.
6. **Power Consumption Sensor:** Watts, for a device (e.g., HVAC, refrigerator).

I'll also include a timestamp for each reading.

Here's how we can simulate the data and the analysis questions:

Simulated IoT Sensor Data (100,000 Lines)

The data will be in a CSV (Comma Separated Values) format. Each row will represent a sensor reading at a specific timestamp.

Columns:

- `Timestamp`: Date and time of the reading (e.g., `YYYY-MM-DD HH:MM:SS`)
- `Temperature_C`: Temperature in Celsius (e.g., 22.5)
- `Humidity_Percent`: Relative humidity in percentage (e.g., 60.2)
- `Light_Lux`: Ambient light in Lux (e.g., 500.1)
- `Motion_Detected`: 0 for no motion, 1 for motion (e.g., 0)
- `Door_Open`: 0 for closed, 1 for open (e.g., 1)
- `Power_Consumption_Watts`: Power consumption in Watts (e.g., 1500.3)

Data Generation Logic:

- **Timestamps:** Will span approximately 70 days (100,000 lines / ~60 readings per hour * 24 hours).
- **Temperature:** Will show daily cycles and some general fluctuations.
- **Humidity:** Will often correlate with temperature, but also have independent variations.
- **Light:** Will show strong daily patterns (day vs. night) and react to motion/door changes.
- **Motion:** Sporadic, but potentially more frequent during certain hours.
- **Door:** Sporadic, representing occasional opening/closing.
- **Power Consumption:** Will have baseline usage, with spikes correlating to temperature (e.g., HVAC turning on) or motion (e.g., lights turning on with motion).

Due to the large size (100,000 lines), I cannot display the entire dataset here. Instead, I will provide a Python script that generates this CSV file. Your students can then use this script to create the data.

```python
Python
import pandas as pd
import numpy as np
from datetime import datetime, timedelta

def generate_iot_data(num_lines=100000):
    start_time = datetime(2025, 1, 1, 0, 0, 0)
    data = []

    for i in range(num_lines):
        # Simulate time passing, roughly every minute
        current_time = start_time + timedelta(minutes=i * np.random.uniform(0.8, 1.2)) # Slight
variability in time step

        # --- Sensor Data Generation ---
        # Temperature (C): Daily cycle with some noise and weekly trend
        base_temp = 20 + 5 * np.sin(current_time.dayofyear * 2 * np.pi / 365) # Yearly cycle
        daily_temp_swing = 3 * np.sin(current_time.hour * np.pi / 12) # Daily cycle
        temperature = base_temp + daily_temp_swing + np.random.normal(0, 0.8)

        # Humidity (%): Correlates somewhat with temperature, but also independent noise
        base_humidity = 60 - 0.5 * (temperature - 20) # Inverse relation to temperature
        humidity = base_humidity + np.random.normal(0, 3)
        humidity = np.clip(humidity, 30, 95) # Keep within realistic bounds

        # Light (Lux): Strong daily cycle (day/night)
        # Higher during day, low at night. Random spikes for motion/door.
        light = 0
        if 6 <= current_time.hour < 20: # Daytime
            light = 200 + 800 * np.sin((current_time.hour - 6) * np.pi / 14) # Peak at noon
            light += np.random.normal(0, 50)
        else: # Nighttime
            light = 5 + np.random.normal(0, 5) # Very low at night

        # Motion Sensor (Binary): Sporadic, higher during active hours
        motion_detected = 0
        if np.random.rand() < 0.05: # Base probability of motion
            if 7 <= current_time.hour < 22 and np.random.rand() < 0.7: # Higher probability during
active hours
                motion_detected = 1
```

```
    # Door/Window Sensor (Binary): Sporadic open/close
    door_open = 0
    if np.random.rand() < 0.01: # Low probability of opening
        door_open = 1
        if np.random.rand() < 0.5 and motion_detected == 1: # More likely to open if motion
            door_open = 1


    # Power Consumption (Watts): Baseline + spikes
    power_consumption = 50 + np.random.normal(0, 10) # Baseline usage

    # HVAC based on temperature
    if temperature > 25:
        power_consumption += 1000 + np.random.normal(0, 100) # AC turns on
    elif temperature < 18:
        power_consumption += 800 + np.random.normal(0, 80) # Heater turns on

    # Lights based on light levels and motion
    if light < 100 and motion_detected == 1:
        power_consumption += 50 + np.random.normal(0, 10) # Lights turn on

    # Random spikes for other devices
    if np.random.rand() < 0.02:
        power_consumption += np.random.uniform(100, 500) # Random appliance usage

    power_consumption = np.clip(power_consumption, 0, 3000) # Keep within realistic bounds


    data.append([
        current_time.strftime('%Y-%m-%d %H:%M:%S'),
        round(temperature, 2),
        round(humidity, 2),
        round(light, 2),
        motion_detected,
        door_open,
        round(power_consumption, 2)
    ])

df = pd.DataFrame(data, columns=[
    'Timestamp', 'Temperature_C', 'Humidity_Percent', 'Light_Lux',
    'Motion_Detected', 'Door_Open', 'Power_Consumption_Watts'
])
return df
```

```
if __name__ == "__main__":
    print("Generating 100,000 lines of IoT sensor data... This may take a moment.")
    df = generate_iot_data(num_lines=100000)
    output_filename = "iot_sensor_data.csv"
    df.to_csv(output_filename, index=False)
    print(f"Data generation complete. Saved to {output_filename}")
    print(f"First 5 rows:\n{df.head()}")
    print(f"Last 5 rows:\n{df.tail()}")
    print(f"Total rows: {len(df)}")
```

Instructions for Students:

1. **Save the code:** Save the provided Python script as `generate_iot_data.py`.
2. **Run the script:** Open a terminal or command prompt, navigate to the directory where you saved the file, and run: `python generate_iot_data.py`
3. **Find the data:** A file named `iot_sensor_data.csv` will be created in the same directory. This is your dataset.

Data Analysis Questions for Students

These questions are designed to guide students through various aspects of data analysis, from basic exploration to more advanced statistical and machine learning concepts.

Part 1: Data Exploration and Cleaning (Fundamental)

1. **Load and Inspect:**
    - Load the `iot_sensor_data.csv` file into a Pandas DataFrame.
    - Display the first 5 and last 5 rows.
    - Check the data types of each column. Are they as expected? If not, how would you convert them (especially `Timestamp`)?
    - Get a summary of the numerical columns (mean, median, min, max, standard deviation) using `df.describe()`. What insights can you gain from this initial summary?
2. **Missing Values:**
    - Check for any missing values in the dataset. If found, how would you handle them (e.g., drop, fill with mean/median/mode, interpolate)? Justify your chosen method.
3. **Time Series Conversion:**
    - Convert the `Timestamp` column to a proper datetime format.

    - Set the `Timestamp` column as the DataFrame index. Why is this useful for time-series data?
4. **Resampling:**
    - Resample the data to an hourly frequency. Calculate the mean for numerical sensors and the sum for binary sensors (motion, door) for each hour.
    - Resample the data to a daily frequency. What trends can you observe?

Part 2: Visualization and Trends (Intermediate)

1. **Sensor Over Time:**
    - Plot the `Temperature_C` and `Humidity_Percent` over time for a 3-day period. Describe the observed patterns (e.g., daily cycles, general trends).
    - Plot `Light_Lux` over time for a 3-day period. How does it correlate with the time of day?
2. **Binary Sensor Analysis:**
    - Calculate the total number of motion detections and door openings.
    - Plot the daily count of `Motion_Detected` and `Door_Open` events. Are there specific days with higher activity?
3. **Power Consumption Analysis:**
    - Plot `Power_Consumption_Watts` over time for a 3-day period. Identify any peaks or unusual patterns.
    - Create a histogram of `Power_Consumption_Watts`. What is the typical power consumption? Are there outliers?
4. **Correlations:**
    - Calculate the correlation matrix for `Temperature_C`, `Humidity_Percent`, `Light_Lux`, and `Power_Consumption_Watts`.
    - Visualize the correlation matrix using a heatmap.
    - Which sensor readings are strongly correlated? Why do you think this is the case? For example, how does temperature relate to power consumption?

Part 3: Advanced Analysis and Prediction (Advanced)

1. **Daily Profiles:**
    - Calculate the average `Temperature_C`, `Humidity_Percent`, `Light_Lux`, and `Power_Consumption_Watts`for each hour of the day across the entire dataset.
    - Plot these average daily profiles. What insights do these profiles offer about typical daily usage and environmental conditions?
2. **Anomaly Detection (Conceptual/Basic):**
    - Define what an "anomaly" might look like for `Power_Consumption_Watts`.
    - Propose a simple method to identify potential anomalies in power consumption (e.g., values exceeding a certain standard deviation from the rolling mean).
    - Plot the `Power_Consumption_Watts` and highlight any detected anomalies.

3. **Event Correlation:**
    - How does `Power_Consumption_Watts` change when `Motion_Detected` is 1 compared to when it's 0, especially during low light conditions? Can you infer any automation (e.g., lights turning on with motion)?
    - Is there a noticeable change in `Temperature_C` or `Humidity_Percent` when `Door_Open` is 1?
4. **Predictive Modeling (Conceptual/Regression):**
    - **Challenge:** Can you predict `Power_Consumption_Watts` based on other sensor readings (e.g., `Temperature_C`, `Light_Lux`, `Motion_Detected`, `Door_Open`, and time-based features like hour of day)?
    - **Steps:**
        - Prepare your data (e.g., feature engineering for time, one-hot encoding for binary sensors if using certain models).
        - Split the data into training and testing sets.
        - Choose a simple regression model (e.g., Linear Regression, Decision Tree Regressor, or a basic scikit-learn model).
        - Train the model and evaluate its performance (e.g., Mean Absolute Error, R-squared).
        - (Optional) Discuss potential challenges and improvements for building a more accurate predictive model.
5. **Classification (Conceptual):**
    - **Challenge:** Can you predict if `Motion_Detected` will be 1 in the next 10 minutes based on historical sensor data and time of day?
    - **Considerations:** This is a more complex time-series prediction problem. Discuss what features you might use and what kind of classification model would be suitable.

Part 4: Open-ended Questions and Discussion

1. **IoT System Design:**
    - What are the limitations of this simulated dataset? What other sensors or data points would be useful to collect for a more comprehensive smart home analysis?
    - How could this sensor data be used to improve energy efficiency in a smart home?
2. **Data Quality:**
    - What are some potential sources of errors or noise in real-world IoT sensor data?
    - How important is data quality in IoT applications, and what are the consequences of poor data quality?
3. **Ethical Considerations:**
    - What privacy concerns might arise from collecting and analyzing this type of sensor data in a real home?
    - How can these concerns be mitigated?