Advanced Combination Questions

1. **Given a DataFrame with `Date`, `Product`, and `Sales`, perform the following:**
2. Convert `Date` to a datetime object.
3. Set `Date` as the index.
4. Resample the data to monthly frequency and calculate the total `Sales` for each month.
5. Plot the time series data.
6. **Merge two DataFrames `df1` and `df2` on a common column `ID` and perform the following:**
7. Handle missing values by filling with the mean of each column.
8. Calculate the correlation between numerical columns in the merged DataFrame.
9. **Create a DataFrame with `User_ID`, `Event_Date`, and `Activity_Level`. Perform the following:**
10. Convert `Event_Date` to datetime.
11. Group by `User_ID` and calculate the total `Activity_Level` per user.
12. Normalize `Activity_Level` across users.
13. **Given a DataFrame with `Product`, `Region`, and `Sales`, perform the following:**
14. Pivot the DataFrame to show `Sales` for each `Product` across different `Regions`.
15. Apply a rolling mean with a window of 3 on the `Sales` data.
16. **Create a DataFrame with hierarchical indexing (`Year`, `Month`, `Day`) and perform the following:**
17. Aggregate the data to get the average `Value` for each `Year`.
18. Reset the index and flatten the multi-index DataFrame.
19. **Use `apply` to perform the following on a DataFrame with `Employee_ID`, `Salary`, and `Years_Experience`:**
20. Apply a function that calculates a performance bonus based on `Years_Experience`.
21. Add a new column `Total_Compensation` which is the sum of `Salary` and `Bonus`.
22. **Given a DataFrame with `Category`, `Date`, and `Revenue`, perform the following:**
23. Resample the data by quarter and calculate the sum of `Revenue`.
24. Plot the quarterly revenue trend for each `Category`.
25. **Create a DataFrame with `Transaction_ID`, `Amount`, `Category`, and `Date`. Perform the following:**
26. Handle missing values in `Amount` using forward fill.
27. Use `groupby` to calculate the average `Amount` by `Category` and `Month`.
28. **Use `pd.merge` to join two DataFrames `orders` and `products` on `Product_ID` and perform the following:**
29. Calculate the total `Order_Value` for each product by multiplying `Quantity` by `Price`.
30. Sort the DataFrame by `Order_Value` in descending order.
31. **Given a DataFrame with `Customer_ID`, `Purchase_Date`, and `Amount`, perform the following:**
    - Calculate the time between each purchase for each customer.
    - Create a column indicating the number of days since the last purchase.

32. **Create a DataFrame with `Employee_ID`, `Hire_Date`, `Salary`, and `Department`. Perform the following:**
    - Calculate the tenure in years for each employee.
    - Group by `Department` and calculate the average `Salary` and `Tenure`.
33. **Given a DataFrame with `Date`, `Sales`, and `Store`, perform the following:**
    - Resample the data to weekly frequency and compute the sum of `Sales`.
    - Identify and flag any outliers in weekly sales using the IQR method.
34. **Use `pd.pivot_table` to create a pivot table from a DataFrame with `Product`, `Region`, and `Sales` to show the total `Sales` per `Region` for each `Product`.**
35. **Create a DataFrame with `Date`, `Temperature`, and `City`. Perform the following:**
    - Convert `Date` to a datetime object.
    - Group by `City` and calculate the average `Temperature` per month.
36. **Given a DataFrame with `Transaction_Date`, `Amount`, and `Customer_ID`, perform the following:**
    - Create a new column indicating the cumulative sum of `Amount` per `Customer_ID`.
    - Calculate the percentage change in cumulative `Amount` for each customer.
37. **Use `applymap` to apply a custom function that formats numerical values in a DataFrame to two decimal places.**
38. **Create a DataFrame with `Date`, `Sales`, `Profit`, and `Region`. Perform the following:**
    - Calculate the monthly profit margin (Profit/Sales) for each `Region`.
    - Plot the monthly profit margin trends for each region.
39. **Given a DataFrame with `Customer_ID`, `Order_Date`, and `Order_Value`, perform the following:**
    - Calculate the average `Order_Value` per month for each `Customer_ID`.
    - Use `pivot_table` to summarize the average monthly order value by `Customer_ID`.
40. **Create a DataFrame with `Item`, `Quantity`, and `Price`. Perform the following:**
    - Create a new column `Total_Cost` that is the product of `Quantity` and `Price`.
    - Calculate the average `Total_Cost` per `Item`.
41. **Given a DataFrame with `Employee_ID`, `Join_Date`, and `Leave_Date`, perform the following:**
    - Calculate the length of service for each employee in days.
    - Create a column indicating if the employee has left the company.
42. **Create a DataFrame with `Product`, `Sales`, and `Date`. Perform the following:**
    - Compute the cumulative sales for each `Product` over time.
    - Use `groupby` to calculate the yearly growth rate of cumulative sales for each product.
43. **Given a DataFrame with `Transaction_ID`, `Amount`, and `Category`, perform the following:**
    - Create a new column that indicates whether each transaction amount is above the average for its category.
    - Group by `Category` and calculate the proportion of transactions above the average.
44. **Use `pd.DataFrame.query` to filter a DataFrame with `Age` and `Salary` to include only rows where `Age` is between 30 and 50 and `Salary` is above the median.**
45. **Create a DataFrame with `Employee_ID`, `Task`, and `Hours_Worked`. Perform the following:**
    - Calculate the total hours worked per employee.

    - Group by `Task` and calculate the average hours worked.

46. **Given a DataFrame with `Date`, `Temperature`, and `Location`, perform the following:**
    - Calculate the monthly average temperature for each `Location`.
    - Plot the temperature trends for each location over time.
47. **Create a DataFrame with `Product`, `Region`, and `Sales`. Perform the following:**
    - Use `pd.crosstab` to create a contingency table showing the count of products sold in each region.
    - Calculate the percentage of total sales contributed by each product.
48. **Given a DataFrame with `Date`, `Sales`, and `Store`, perform the following:**
    - Resample the data to annual frequency and calculate the total `Sales` per store.
    - Use `groupby` to compute the store with the highest total sales each year.
49. **Use `pd.merge` to combine DataFrames with `Customer_ID` and `Order_Date`. Perform the following:**
    - Calculate the time difference between each customerâ€™s first and last order.
    - Group by `Customer_ID` and compute the average order frequency.
50. **Create a DataFrame with `Date`, `Stock_Price`, and `Volume`. Perform the following:**
    - Compute the daily percentage change in `Stock_Price`.
    - Calculate the rolling standard deviation of `Volume` over a 30-day window.
51. **Given a DataFrame with `Date`, `Temperature`, and `City`, perform the following:**
    - Use `pd.Grouper` to group data by week and calculate the weekly average temperature.
    - Identify and visualize the cities with the highest and lowest average weekly temperatures.