Advanced Combination Pandas Questions

1. **Given a DataFrame with `Date`, `Category`, and `Value`, perform the following:**
2. Convert `Date` to a datetime object.
3. Create a new column indicating the month and year.
4. Group by `Category` and calculate the mean `Value` for each month.
5. **Create a DataFrame with `Product`, `Price`, and `Quantity_Sold`. Perform the following:**
6. Compute the total revenue per product (i.e., `Price` Ã— `Quantity_Sold`).
7. Sort the DataFrame by total revenue in descending order.
8. **Given a DataFrame with `User_ID`, `Event_Date`, and `Action`, perform the following:**
9. Convert `Event_Date` to a datetime object.
10. Calculate the frequency of each `Action` per user.
11. Create a heatmap showing action frequency over time.
12. **Use `pd.merge` to combine DataFrames `sales` and `targets` on `Product_ID`. Perform the following:**
13. Calculate the difference between `Sales` and `Target` for each product.
14. Identify products where actual sales exceed the target.
15. **Create a DataFrame with hierarchical indexes (`Department`, `Employee_ID`) and columns `Salary`, `Joining_Date`. Perform the following:**
16. Calculate the average salary within each department.
17. Compute the tenure in years for each employee and add it as a column.
18. **Given a DataFrame with `Date`, `Sales`, `Region`, and `Store`, perform the following:**
19. Resample the data to monthly frequency and calculate the total `Sales` for each `Store`.
20. Use `pd.crosstab` to analyze sales distribution across `Regions`.
21. **Create a DataFrame with `Transaction_ID`, `Amount`, `Date`, and `Customer_ID`. Perform the following:**
22. Calculate the cumulative sum of `Amount` per customer.
23. Determine the total amount spent by each customer in each year.
24. **Given a DataFrame with `Product`, `Sales`, `Cost`, and `Date`, perform the following:**
25. Calculate the profit margin (i.e., `(Sales - Cost) / Sales`).
26. Resample the data to weekly frequency and compute the average profit margin.
27. **Use `apply` to transform a DataFrame with `Employee_ID`, `Hours_Worked`, and `Task`. Perform the following:**
28. Create a function that categorizes `Hours_Worked` into `Low`, `Medium`, and `High`.
29. Apply this function to add a new column `Workload_Category`.
30. **Create a DataFrame with `Date`, `Temperature`, and `City`. Perform the following:**
    - Calculate the rolling average of `Temperature` over a 7-day window for each city.
    - Plot the rolling average temperatures for all cities.

31. **Given a DataFrame with `Date`, `Sales`, `Store`, and `Region`, perform the following:**
    - Group by `Store` and calculate the monthly growth rate of `Sales`.
    - Identify stores with the highest growth rates.
32. **Use `pd.pivot_table` to analyze a DataFrame with `Product`, `Region`, `Sales`, and `Profit`. Perform the following:**
    - Create a pivot table showing total `Sales` and average `Profit` by `Region` and `Product`.
    - Handle missing values in the pivot table by filling with zeros.
33. **Create a DataFrame with `User_ID`, `Activity_Date`, and `Activity_Level`. Perform the following:**
    - Calculate the average `Activity_Level` per month for each user.
    - Identify users with the highest average activity levels.
34. **Given a DataFrame with `Date`, `Stock_Price`, and `Volume`, perform the following:**
    - Calculate the daily return (percentage change) of `Stock_Price`.
    - Compute the rolling volatility of `Stock_Price` over a 30-day window.
35. **Use `pd.DataFrame.query()` to filter a DataFrame with `Age`, `Salary`, and `Department`. Perform the following:**
    - Select rows where `Age` is between 25 and 45 and `Salary` is above the median salary for the department.
36. **Create a DataFrame with `Product`, `Sales`, `Cost`, and `Date`. Perform the following:**
    - Calculate the profit for each product and add it as a new column.
    - Group by `Product` and calculate the total profit.
37. **Given a DataFrame with `Date`, `Sales`, `Store`, and `Region`, perform the following:**
    - Resample the data to quarterly frequency and calculate the total `Sales` per `Region`.
    - Identify regions with the highest total sales.
38. **Use `pd.merge` to combine DataFrames with `Employee_ID` and `Performance_Score`. Perform the following:**
    - Add a column indicating if an employee's performance score is above the median.
    - Calculate the average performance score by department.
39. **Create a DataFrame with `Transaction_Date`, `Amount`, `Customer_ID`, and `Category`. Perform the following:**
    - Calculate the total amount spent by each customer per category.
    - Visualize the spending distribution by category for each customer.
40. **Given a DataFrame with `Date`, `Temperature`, and `City`, perform the following:**
    - Calculate the monthly average temperature for each city.
    - Plot the temperature trends for the top 3 cities with the highest average monthly temperature.
41. **Use `pd.DataFrame.aggregate()` to perform multiple aggregation functions on a DataFrame with `Region`, `Sales`, and `Profit`. Perform the following:**
    - Calculate the total and average `Sales` and `Profit` for each `Region`.
42. **Create a DataFrame with `Date`, `Event`, `Category`, and `Value`. Perform the following:**
    - Pivot the DataFrame to show `Value` aggregated by `Category` and `Date`.
    - Use `pd.melt` to convert the pivoted DataFrame back to long format.
43. **Given a DataFrame with `Product`, `Sales`, and `Price`, perform the following:**

- Calculate the average `Price` and `Sales` per product.
- Identify the top 5 products by total sales.

44. **Use `apply` to perform a complex transformation on a DataFrame with `User_ID`, `Transaction_Date`, and `Amount`. Perform the following:**

    - Create a custom function to calculate the average transaction amount per user over the last 30 days.
    - Apply this function and add it as a new column.

45. **Create a DataFrame with `Employee_ID`, `Start_Date`, `End_Date`, and `Project`. Perform the following:**

    - Calculate the duration of each project in days.
    - Group by `Project` and compute the average project duration.

46. **Given a DataFrame with `Date`, `Sales`, `Store`, and `Region`, perform the following:**

    - Calculate the year-over-year growth rate for each store.
    - Identify stores with the highest growth rates over the past year.

47. **Use `pd.DataFrame.rolling()` to apply a rolling function on a DataFrame with `Date`, `Sales`, and `Store`. Perform the following:**

    - Calculate a 7-day rolling average of `Sales` for each store.
    - Plot the rolling averages for all stores.

48. **Create a DataFrame with `Product`, `Sales`, `Cost`, and `Date`. Perform the following:**

    - Compute the profit margin (i.e., `(Sales - Cost) / Sales`).
    - Group by `Product` and calculate the average profit margin.

49. **Given a DataFrame with `Date`, `Sales`, `Store`, and `Region`, perform the following:**

    - Resample the data to yearly frequency and compute the total `Sales` per store.
    - Use `groupby` to calculate the store with the highest sales each year.

50. **Use `pd.DataFrame.query()` to filter a DataFrame with `Product`, `Sales`, and `Date`. Perform the following:**

    - Select rows where `Sales` are above a specified threshold and the `Date` is within the last 6 months.