DataFrame Creation and Basic Operations

1. **Create a DataFrame from a list of dictionaries with keys `ID`, `Name`, and `Age`. Display the DataFrame.**
2. **Convert a NumPy array of shape (3, 4) into a Pandas DataFrame with column names `A`, `B`, `C`, and `D`.**
3. **Create a DataFrame from a CSV file named `sales_data.csv` and display the last 5 rows.**
4. **Construct a DataFrame from a list of lists where each sublist represents a row. Assign column names `Product`, `Quantity`, and `Price`.**
5. **Read an Excel file named `financials.xlsx` into a DataFrame. Display the first 3 rows.**

Data Selection and Filtering

1. **Select rows where the value in the `Salary` column is less than 60000.**
2. **Filter a DataFrame to include only rows where the `Department` column is `'Sales'` and `Age` is greater than 30.**
3. **Select specific columns `Name` and `Age` from a DataFrame and display them.**
4. **Use `.loc` to select rows where the `Score` is between 50 and 75 inclusive.**
5. **Use `.iloc` to select the first 5 rows and columns 2 to 4 of a DataFrame.**

Data Manipulation

1. **Add a new column `Experience` to a DataFrame with values calculated as the difference between the current year and the `Year_Joined` column.**
2. **Remove rows with duplicate values in the `Email` column.**
3. **Create a new column `Total` by multiplying the `Quantity` and `Price` columns.**
4. **Rename the columns of a DataFrame from `A1`, `A2`, `A3` to `B1`, `B2`, `B3`.**
5. **Replace all occurrences of `NaN` in a DataFrame with the median value of their respective columns.**

Data Aggregation

1. **Group a DataFrame by `Department` and calculate the average `Salary` for each department.**
2. **Compute the total sales for each product category using a DataFrame with columns `Product` and `Sales`.**
3. **Create a pivot table to show the average `Sales` for each `Region` and `Quarter`.**
4. **Calculate the sum of `Quantity` for each `Product` using a DataFrame.**
5. **Find the maximum value in the `Price` column and identify the corresponding row.**

Time Series Data

1. **Create a DataFrame with a `Date` column and set it as the index. Display the data for the year 2023.**
2. **Resample a time series DataFrame to show monthly data and calculate the mean of `Sales` for each month.**
3. **Create a DataFrame with `Date` and `Sales` columns. Add a new column that shows the rolling average of `Sales` with a window of 7 days.**
4. **Convert a `Date` column in a DataFrame from string format to datetime format.**

5. **Plot a time series of `Sales` data from a DataFrame using Pandas built-in plotting functions.**

Advanced Data Operations

1. **Merge two DataFrames on a common column `Employee_ID` using an inner join.**
2. **Concatenate two DataFrames along the columns and reset the index.**
3. **Perform a left join between two DataFrames on `Customer_ID` and display the result.**
4. **Use the `query` method to select rows where `Sales` is greater than the mean `Sales` of the DataFrame.**
5. **Apply a custom function to each row in a DataFrame that calculates a new value based on existing columns.**

Data Cleaning

1. **Identify and fill missing values in a DataFrame with the forward fill method.**
2. **Use the `dropna` method to remove rows with any missing values in a DataFrame.**
3. **Replace specific values in a column with new values (e.g., replace `'Unknown'` with `'N/A'`).**
4. **Detect outliers in the `Score` column using the IQR method.**
5. **Standardize the values in a numerical column to have a mean of 0 and a standard deviation of 1.**

Data Visualization

1. **Plot a histogram of the `Age` column from a DataFrame using Pandas plotting functions.**
2. **Create a scatter plot of `Quantity` vs `Price` from a DataFrame.**
3. **Generate a bar plot showing the average `Sales` for each `Region`.**
4. **Plot a box plot of `Salary` grouped by `Department`.**
5. **Create a line plot showing the trend of `Revenue` over time.**

Miscellaneous

1. **Convert a DataFrame into a dictionary with `orient='records'`.**
2. **Use the `sample` method to randomly select 5 rows from a DataFrame.**
3. **Save a DataFrame to an Excel file named `output.xlsx`.**
4. **Convert a DataFrame to a NumPy array and display its shape.**
5. **Check the data types of all columns in a DataFrame and convert a specific column to `float`.**
6. **Use the `melt` function to unpivot a DataFrame from wide to long format.**
7. **Create a new DataFrame by selecting rows from an existing DataFrame based on a condition in another DataFrame.**
8. **Get the count of unique values in the `Category` column of a DataFrame.**
9. **Calculate the correlation matrix of numerical columns in a DataFrame.**
10. **Create a DataFrame with `Name`, `Age`, and `Score` columns and save it as a JSON file.**