For all Pandas tasks, use **New York City Airbnb Open Data** from Kaggle.

It is available at the link after registration.

New York City Airbnb Open Data

Practical Task 1: Basic Data Exploration and Cleaning with Pandas

Objective:

Introduce basic data exploration and data cleaning techniques using the New York City Airbnb Open Data dataset.

Requirements:

1. **Data Loading and Initial Inspection:**
   - Load the New York City Airbnb Open Data dataset into **DataFrame**.
   - Inspect the first few rows of the dataset using the **head()** method to understand its structure.
   - Retrieve basic information about the dataset, such as the number of entries, column data types, and memory usage, using the **info()** method.
2. **Handling Missing Values:**
   - Identify columns with missing values and count the number of missing entries per column.
   - Handle missing values in the **name**, **host_name**, and **last_review** columns:
     - For **name** and **host_name**, fill missing values with the string **"Unknown"**.
     - For **last_review**, fill missing values with a special value "**NaT"**. "**NaT"** stands for Not a Time.
3. **Data Transformation:**
   - ***Categorize Listings by Price Range:***
     Create a new column **price_category** that categorizes listings into different price ranges, such as **Low**, **Medium**, **High**, based on defined thresholds (e.g., Low: price < $100, Medium: $100 <= price < $300, High: price >= $300).
   - ***Create a length_of_stay_category column:***
     Categorize listings based on their **minimum_nights** into **short-term**, **medium-term**, and long-term stays.
     For example, **short-term** might be minimum_nights <= 3, **medium-term** minimum_nights between 4 and 14, and **long-term** minimum_nights > 14.
4. **Data Validation:**
   - Verify that the data transformations and cleaning steps were successful by re-inspecting the **DataFrame**.

- Ensure that the dataset has no missing values in critical columns (**name**, **host_name**, **last_review**).
- Confirm that all **price** values are greater than 0. If you find rows with price equal to 0, remove them.

5. **Output Function:**
   - Implement a separate function named **print_dataframe_info** that takes the **DataFrame** and an optional message as input, prints the message, and displays basic information about the **DataFrame** (e.g., number of entries, missing values).
   - Use this function to output the state of the **DataFrame** before and after cleaning.

6. **Execution and Verification:**
   Test the script to ensure that all data loading, cleaning, and transformation operations are executed correctly. Validate that the cleaned dataset is ready for further analysis.

Deliverables: A Python script (.py file) containing all the functions and code to load the dataset, perform the cleaning and transformation operations, and output the results using the **print_dataframe_info** function.

Save the cleaned dataset as a new CSV file named **cleaned_airbnb_data.csv** for use in subsequent tasks.

Practical Task 2: Data Selection, Filtering, and Aggregation with Pandas

Objective:

Develop skills in selecting, filtering, and aggregating data within the New York City Airbnb Open Data dataset.

Requirements:

1. **Data Selection and Filtering:**
   - Use **.iloc** and **.loc** to select specific rows and columns based on both position and labels.
   - Filter the dataset to include only listings in specific neighborhoods (e.g., Manhattan, Brooklyn).
   - Further filter the dataset to include only listings with a **price** greater than $100 and a **number_of_reviews** greater than 10.
   - Select columns of interest such as **neighbourhood_group**, **price**, **minimum_nights**, **number_of_reviews**, **price_category** and **availability_365** for further analysis.

2. **Aggregation and Grouping:**

- Group the filtered dataset by **neighbourhood_group** and **price_category** to calculate aggregate statistics:
    - Calculate the average **price** and **minimum_nights** for each group.
    - Compute the average **number_of_reviews** and **availability_365** for each group to understand typical review counts and availability within each neighborhood and price category.

3. **Data Sorting and Ranking:**
    - Sort the data by **price** in descending order and by **number_of_reviews** in ascending order.
    - Create a ranking of neighborhoods based on the total number of listings and the average **price**.
4. **Output:**
    - Implement a separate function named **print_grouped_data** that takes the grouped **DataFrame** and an optional message as input, prints the message, and displays the grouped data.
    - Output the results of your aggregations and rankings.
5. **Execution and Verification:**
    - Test the script to ensure that all selection, filtering, and aggregation operations are executed correctly.
    - Validate that the filtered and grouped dataset is accurate and provides meaningful insights.

Deliverables:

A Python script (.py file) containing all the functions and code to perform the data selection, filtering, aggregation, and ranking operations, and to output the results using the **print_grouped_data** function.

Save the aggregated data as a new CSV file named **aggregated_airbnb_data.csv**.

Practical Task 3: Advanced Data Manipulation, Descriptive Statistics, and Time Series Analysis with Pandas

Objective: Enhance data manipulation skills while generating actionable business insights by performing advanced transformations, descriptive statistics, and time series analysis on the New York City Airbnb Open Data dataset.

Requirements:

1. **Advanced Data Manipulation:**

- **Analyze Pricing Trends Across Neighborhoods and Room Types:**
    - Use the **pivot_table** function to create a detailed summary that reveals the average price for different combinations of **neighbourhood_group** and **room_type**. This analysis will help identify high-demand areas and optimize pricing strategies across various types of accommodations (e.g., Entire home/apt vs. Private room).
- **Prepare Data for In-Depth Metric Analysis:**
    - Transform the dataset from a wide format to a long format using the **melt** function. This restructuring facilitates more flexible and detailed analysis of key metrics like **price** and **minimum_nights**, enabling the identification of trends, outliers, and correlations.
- **Classify Listings by Availability:**
    - Create a new column **availability_status** using the **apply** function, classifying each listing into one of three categories based on the **availability_365** column:
        - **"Rarely Available"**: Listings with fewer than 50 days of availability in a year.
        - **"Occasionally Available"**: Listings with availability between 50 and 200 days.
        - **"Highly Available"**: Listings with more than 200 days of availability.
    - Analyze trends and patterns using the new **availability_status** column, and investigate potential correlations between availability and other key variables like **price**, **number_of_reviews**, and **neighbourhood_group** to uncover insights that could inform marketing and operational strategies.

2. **Descriptive Statistics:**
    - Perform basic descriptive statistics (e.g., **mean**, **median**, **standard deviation**) on numeric columns such as **price**, **minimum_nights**, and **number_of_reviews** to summarize the dataset's central tendencies and variability, which is crucial for understanding overall market dynamics.
3. **Time Series Analysis:**
    - **Convert and Index Time Data:**
        - Convert the **last_review** column to a datetime object and set it as the index of the DataFrame to facilitate time-based analyses.
    - **Identify Monthly Trends:**

        - Resample the data to observe monthly trends in the number of reviews and average prices, providing insights into how demand and pricing fluctuate over time.
    - **Analyze Seasonal Patterns:**
        - Group the data by month to calculate monthly averages and analyze seasonal patterns, enabling better forecasting and strategic planning around peak periods.

4. **Output Function:**
    - Implement a separate function named **print_analysis_results** that takes the DataFrame or Series and an optional message as input, prints the message, and displays the results of your analysis. This function should be used to clearly present the findings from your descriptive statistics and time series analysis.
5. **Execution and Verification:**
    - Test the script to ensure that all data manipulation, statistical, and time series operations are executed correctly.
    - Validate that the results provide meaningful insights, are accurate, and are ready for visualization or further analysis, ensuring the data-driven decisions are well-supported.

Deliverables:

- A Python script (.py file) containing all the functions and code necessary to perform the advanced data manipulation, descriptive statistics, and time series analysis, and to output the results using the **print_analysis_results** function.
- Save the results of your time series analysis as a new CSV file named time_series_airbnb_data.csv.