# Airbnb NYC Data Cleaning and Exploratory Analysis – Module Summary

## Overview

This project builds a reproducible data-cleaning and exploratory analysis workflow for the **AB_NYC_2019** Airbnb dataset, which contains listings from New York City. The notebook loads the raw CSV file, applies a sequence of cleaning functions, and then performs exploratory data analysis (EDA) with summary statistics and visualizations. The goal is to demonstrate practical data-wrangling, visualization, and interpretation on a real-world dataset.

## Dataset Description

The **AB_NYC_2019** dataset represents Airbnb listings in New York City, including information such as listing name, host, neighbourhood, room type, price, minimum nights, and review activity. It contains tens of thousands of rows and around a dozen columns, giving a reasonably detailed snapshot of the NYC short-term rental market.

For this project, I focused primarily on variables related to **price**, **neighbourhood_group**, **room_type**, **minimum_nights**, and **reviews_per_month**, because these fields are directly tied to cost, location, and guest activity. Other columns, such as latitude/longitude and textual descriptions, were not explored in depth but could support future spatial or NLP-based analyses. The dataset is publicly available on Kaggle as **“New York City Airbnb Open Data (AB_NYC_2019)”**:

https://www.kaggle.com/datasets/dgomonov/new-york-city-airbnb-open-data

## Workflow Description (High Level)

### Ingestion

- Loaded the CSV file directly from disk using `pandas.read_csv`, creating a base `DataFrame` called `df`.

- Verified that the data loaded successfully by printing the head of the dataset and inspecting column names.

### Cleaning

- Implemented several reusable cleaning functions (e.g., `drop_missing_essential`, `remove_price_outliers`, `remove_extreme_minimum_nights`, `fill_missing_reviews_per_month`) to manage missing values and outliers.

- Applied these functions in sequence to produce a cleaned dataset `df_clean`.

- This functional approach follows recommendations for modular, testable steps in reproducible Python workflows (Reproducible Data Science with Python: An Open Learning Resource, n.d.).

### Exploratory Analysis

- Created EDA helper functions (`eda_summary`, `eda_price_by_neighbourhood`) to print dataset shape, column types, summary statistics, and grouped aggregations.

- Focused on understanding distributions of prices and how they vary across neighbourhood groups and room types.

### Visualizations

- Produced three main plots using Matplotlib/Seaborn:

1. A **histogram** of listing prices (Figure 1).

2. A **bar chart** of average price by neighbourhood group (Figure 2).

3. A **scatter plot** of price versus minimum nights (Figure 3).

- Each visualization includes clear titles and axis labels to support interpretability and communication best practices (Wickham, 2014).

### Summary

- Concluded with a written summary and interpretation section, capturing what was learned, key patterns, limitations, and open questions.

- The notebook thus forms a compact but complete workflow: ingest → clean → explore → visualize → interpret.

## Key Decisions and Assumptions

### Cleaning Choices

- **Dropping rows with missing essential fields**: I treated `price`, `room_type`, and `neighbourhood_group` as essential variables and removed rows missing any of these values. This decision ensures that all subsequent summaries and plots are based on complete and interpretable records for these key fields, aligning with common recommendations to handle missing data explicitly rather than silently propagating `NaN`s through analysis (Reproducible Data Science with Python: An Open Learning Resource, n.d.).

- **Filtering unrealistic prices**: I restricted prices to a reasonable range (e.g., between 10 and 1000 USD) to remove obvious outliers that could distort histograms and means. This is a simple but practical approach to outlier handling in exploratory work, with the assumption that extremely low or extremely high values are likely to be data entry errors or atypical cases that are not the focus of this project.

- **Capping extreme minimum nights**: I removed listings with very large `minimum_nights` values (e.g., above 365), under the assumption that such listings behave more like long-term rentals than short-term stays. This makes the dataset more consistent with typical Airbnb use cases.

### Focus in EDA

- Focused EDA on questions about **pricing** and **location** because these are primary drivers of user decisions and business strategy in short-term rentals.

- Grouped summaries by `neighbourhood_group` and `room_type` were chosen to highlight structural differences between parts of the city and between types of listings.

### Design of Plots

- **Figure 1 (Histogram of prices)** was designed to show the overall price distribution and highlight skewness. A histogram with many bins plus a kernel density estimate helps visualize both the bulk of typical listings and the long right tail.

- **Figure 2 (Average price by neighbourhood group)** was chosen as a bar chart because it allows straightforward comparison of mean prices for a small number of categories, following standard visualization practice for aggregated categorical comparisons (Wickham, 2014).

- **Figure 3 (Price vs minimum nights)** uses a scatter plot to examine the relationship between stay length and price, showing whether higher prices cluster at particular minimum-night requirements. Alpha blending was used to reduce overplotting and make dense regions more readable, a common best practice when visualizing many overlapping points.

These decisions aim to balance simplicity and interpretability while maintaining transparency about how the data was filtered and summarized.

## Results and Interpretation

The analysis reveals several key patterns:

- **Price distribution** (Figure 1) is heavily right-skewed: most listings cluster at relatively low to moderate nightly rates, with a small number of very expensive properties extending the tail. This suggests that while luxury listings exist, the majority of the market is more budget-friendly or mid-range.

- **Average prices by neighbourhood group** (Figure 2) show clear geographic differences in cost. Certain neighbourhood groups (for example, central or highly touristic areas) tend to have higher average prices, while others appear more affordable. This aligns with expectations that location is a major driver of Airbnb pricing.

- **Price versus minimum nights** (Figure 3) indicates that there is no simple linear relationship between length of stay requirements and nightly price. While some very high minimum-night requirements correspond to distinct pricing patterns, overall the scatter remains noisy, suggesting that host pricing strategies depend on a mix of factors beyond just minimum nights.

- The grouped EDA table of average price by `neighbourhood_group` and `room_type` reinforces these visual impressions by quantifying which combinations (e.g., entire home/apartment in certain neighbourhoods) tend to be the most expensive.

Overall, the results provide a concise but informative picture of how prices vary across NYC's Airbnb landscape and where potential "hot spots" of higher pricing might exist.

## Responsible Practice (Bias and Data Quality)

Several aspects of the workflow could introduce bias or affect data quality:

- ****Removal of records with missing values**** in essential columns can bias the dataset if missingness is systematic (for example, cheaper listings or specific neighbourhoods being less completely documented). This may skew estimates of average price or availability.

- ****Outlier filtering**** based on fixed thresholds (e.g., price between 10 and 1000 USD) assumes that values outside this range are errors or irrelevant, which may not always be true. This choice can underestimate the true variability in the market and downplay luxury segments.

- ****Focusing on a subset of variables**** (price, neighbourhood_group, room_type) means that other important factors such as host practices, review scores, or availability patterns remain unexplored, which could limit understanding of underlying drivers.

To reduce these risks, future work could:

- Compare cleaned results to analyses on the full dataset to assess sensitivity to cleaning choices.

- Use more principled methods for outlier detection (e.g., robust statistics or domain-informed thresholds).

- Perform stratified checks to see whether specific groups (neighbourhoods or room types) are disproportionately affected by data removal.

## Reproducibility

Reproducibility was a key design consideration in this project, in line with recommendations for modern data science workflows (Reproducible Data Science with Python: An Open Learning Resource, n.d.). Key steps include:

- **Environment capture via `requirements.txt`**: I recorded the Python package versions used in the project with:

  ```bash
  pip freeze > requirements.txt
  ```

  Another user can recreate the environment with:

  ```bash
  pip install -r requirements.txt
  ```

- **Single, linear notebook workflow**: The notebook is organized in a top-to-bottom order: introduction → imports → ingestion → cleaning functions → cleaned dataset → EDA functions → visualizations → summary. This reduces hidden state and makes it easier to re-run the entire analysis from a fresh kernel.

- **Functional structure for cleaning and EDA**: By encapsulating cleaning and EDA logic into small, named functions, the workflow becomes easier to reuse, test, and reason about. This modular approach is consistent with guidance on writing reusable analysis code in reproducible Python projects (Reproducible Data Science with Python: An Open Learning Resource, n.d.).

- **Version control (Git) considerations**: While this summary does not include a full Git history, a recommended approach would be to:

  - Maintain a `main` branch with stable versions of the notebook.

  - Use feature branches for larger changes (e.g., adding new cleaning steps or visualizations).

  - Commit frequently with descriptive messages (e.g., "Add price outlier filter" or "Create price vs minimum nights scatter plot").

  This style of workflow helps track changes over time and supports collaborative and reproducible work.

## Sources and Citations (Required)

In-text citations above reference the following sources (fill in full details according to your citation style):

- **Reproducible Data Science with Python: An Open Learning Resource.** (n.d.). Peer-reviewed article on reproducible data science.

- Wickham, H. (2014). *Tidy Data*. *Journal of Statistical Software, 59*(10), 1–23.

You may optionally add other reputable sources, such as:

- Official documentation for Python, pandas, Matplotlib, or Seaborn.

- University lecture notes or textbooks discussing missing data handling, visualization best practices, or reproducible workflows.