THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

THE GEORGE WASHINGTON UNIVERSITY

Airbnb Insights: Uncovering Accommodation Trends in New York City

Author: Smit Snehal Pancholi

Instructor: Dr. Reza Jafari

Date: 04/26/2024

Course Name: Visualization of Complex Data

Course Number: DATS 6401

Table of Contents

Table of figures and tables.

Abstract

This study presents a comprehensive analysis of Airbnb listings in New York City, leveraging a dataset comprising 102,600 properties. Through meticulous examination of property attributes, pricing dynamics, host profiles, and guest feedback, intricate patterns and emerging trends within the local Airbnb market are unveiled. The research aims to illuminate the preferences and behaviors of both guests and hosts in one of the world's premier travel destinations, enriching our understanding of contemporary hospitality trends. By providing actionable guidance for travelers and hosts navigating the dynamic landscape of the NYC Airbnb market, this study contributes to informed decision-making in the accommodation sector. The analysis encompasses dataset description, pre-processing steps, outlier detection, principal component analysis (PCA), normality testing, statistical analyses, data visualization, and the implementation of a dashboard for real-time exploration. Through these efforts, stakeholders gain valuable insights into the nuances of the NYC Airbnb landscape, facilitating informed choices and enhancing the overall guest and host experience. The dashboard is implemented using the Dash framework and deployed on Google Cloud Platform (GCP) for accessibility and usability.

Introduction

Airbnb has become synonymous with modern travel since its establishment in 2008, originally named AirBedandBreakfast.com [1], transforming how people discover and reserve accommodations. Originating in San Francisco, the platform has expanded globally, offering a unique approach to lodging through the sharing economy. Unlike conventional hotels, Airbnb allows individuals to rent out their private spaces, ranging from single rooms to entire homes, democratizing travel by enabling property owners to become hosts and offering travelers a personalized "home away from home" experience. Airbnb facilitates these connections through a seamless online and mobile app interface, generating revenue through a commission system from each transaction.

The vibrancy of New York City—with its iconic skyline, diverse neighborhoods, and status as a cultural and financial hub—reflects the dynamic offerings of Airbnb. The platform has gained significant traction in the city, making NYC a compelling case study for examining Airbnb's impact on urban travel. Analyzing Airbnb's data within the context of New York provides insights into its broader influence on the accommodation sector.

This report employs various analytical methods to decode the intricacies of the New York Airbnb marketplace. It aims to dissect the factors influencing a listing's appeal, analyze pricing models across neighborhoods, and evaluate sought-after features that give certain listings a competitive advantage. By correlating these elements with guest reviews and host data, the study articulates the key determinants of success for Airbnb hosts in NYC and offers insights for guests selecting accommodations.

The report will unfold a sequence of methodical analyses, including but not limited to:

Data Preprocessing and Feature Engineering: Refining the dataset to improve input quality for analysis.

Exploratory Data Analysis (EDA): Surveying the landscape of Airbnb offerings in New York and identifying emerging patterns.

Statistical Testing: Implementing normality tests and employing Pearson correlation metrics to understand relationships between factors such as location, price, and guest satisfaction.

Data Visualization: Crafting plots to illustrate data trends and distributions, enhancing accessibility and engagement.

Dimensionality Reduction: Conducting Principal Component Analysis (PCA) to distill complex data sets into key elements for focused analysis.

Outlier Detection: Identifying and addressing anomalies to ensure data validity and reliability.

Interactive Dashboard Creation: Utilizing the Dash framework to build an interactive tool for real-time data exploration by end-users.

The report will present these findings in a detailed yet accessible format, connecting Airbnb's operational data with the broader economic and cultural landscape of New York City. The conclusion will synthesize insights gleaned from the data, providing a comprehensive overview of Airbnb's impact on NYC's accommodation space and implications for hosts, guests, and the local tourism industry.

Through this report, stakeholders will gain a deeper understanding of the interplay between individual property offerings and overarching travel trends. The analysis aims to offer guidance for informed decision-making, whether for optimizing listings or selecting the ideal New York City experience.

Dataset Description

The dataset in question, designed for a detailed exploration of Airbnb's New York City listings, is comprised of 102,600 entries [2], each entry encapsulating a unique accommodation space available for booking. With a selection of both numerical and categorical variables, the dataset provides a robust framework for statistical analysis and industry insights.

Numerical columns, which are integral for quantitative analysis, include the 'id' and 'host_id' serving as unique identifiers for listings and hosts respectively. The geographical coordinates are denoted by 'lat' and 'long', which are crucial for spatial analysis. Financial elements are captured in 'price' and 'service fee', vital for pricing strategy evaluation. The 'minimum nights', 'number of reviews', 'reviews per month', 'review rate number', and 'availability 365' are quantitative measures that offer insight into listing popularity and market demand.

Categorical variables such as 'neighbourhood group', 'neighbourhood', 'instant_bookable', 'cancellation_policy', and 'room type' offer a qualitative perspective on the listings. These factors encompass location classification, booking and cancellation policies, type of accommodations provided, and regulatory compliance — all of which are vital for understanding market segmentation and regulatory aspects.

The primary columns serving as dependent variables in this research are 'price' and 'number of reviews', as they represent outcomes influenced by various factors in the dataset. Independent variables include 'room type', 'minimum nights', and 'neighbourhood group', among others, which are expected to exert significant effects on the pricing and popularity of listings.

Understanding the interactions between these variables is of paramount importance to the industry. It allows for the optimization of pricing strategies, enhancement of customer satisfaction, and alignment of supply with demand. In an urban environment as dynamic as New York City, where the hospitality industry is fiercely competitive and highly regulated, this dataset is a goldmine. It not only aids businesses in refining their market approach but also assists policymakers in understanding the effects of shared economy platforms on traditional lodging sectors. Through predictive modeling, trend analysis, and market segmentation, this dataset will facilitate a deeper comprehension of Airbnb's influence on travel, tourism, and urban economics.

Data Preprocessing

The data cleaning process was executed with meticulous attention to ensure that the dataset was primed for a thorough analysis. The initial step involved addressing null values, a task which entailed an assessment and removal of such entries to safeguard the dataset's integrity. Further refinement included the elimination of columns deemed irrelevant to the study's goals, specifically 'license' and 'house_rules'. This helped pare down the dataset to its most essential elements. Duplicate records were also identified and purged, a crucial measure to avoid distortion of the study's findings.

	id	NAME	host id	\
0	1001254	Clean & quiet apt home by the park	80014485718	
1	1002102	Skylit Midtown Castle	52335172823	
2	1002403	THE VILLAGE OF HARLEM....NEW YORK !	78829239556	
3	1002755	NaN	85098326012	
4	1003689	Entire Apt: Spacious Studio/Loft by central park	92037596077	

	host_identity_verified	host name	neighbourhood group	neighbourhood	\
0	unconfirmed	Madaline	Brooklyn	Kensington	
1	verified	Jenna	Manhattan	Midtown	
2	NaN	Elise	Manhattan	Harlem	
3	unconfirmed	Garry	Brooklyn	Clinton Hill	
4	verified	Lyndon	Manhattan	East Harlem	

	lat	long	country	country code	instant_bookable	\
0	40.64749	-73.97237	United States	US	False	
1	40.75362	-73.98377	United States	US	False	
2	40.80902	-73.94190	United States	US	True	
3	40.68514	-73.95976	United States	US	True	
4	40.79851	-73.94399	United States	US	False	

Figure 1. Before Preprocessing.

The renaming of columns was conducted systematically to facilitate ease of reference and manipulation during analysis. This involved converting all column names to lowercase and replacing spaces with underscores for uniformity and to align with programming conventions. Pertinent financial columns, 'price' and 'service_fee', underwent cleansing to remove any non-numeric characters, followed by conversion to numeric types to ready them for quantitative analysis.

Standardization extended to the textual data within the 'neighbourhood_group' column, where misspellings were corrected to ensure accuracy in categorization. These preliminary steps were foundational, eliminating discrepancies and enhancing the overall structure of the dataset.

	id	name	host_id	host_identity_verified	host_name	neighbourhood_group	neighbourhood	lat	long	country	country_code	instant_bookable
0	1001254	Clean & quiet apt home by the park	80014485718	unconfirmed	Madaline	Brooklyn	Kensington	40.64749	-73.97237	United States	US	False
1	1002102	Skylit Midtown Castle	52335172823	verified	Jenna	Manhattan	Midtown	40.75362	-73.98377	United States	US	False
4	1003689	Entire Apt: Spacious Studio/Loft by central park	92037596077	verified	Lyndon	Manhattan	East Harlem	40.79851	-73.94399	United States	US	False
5	1004098	Large Cozy 1 BR Apartment In Midtown East	45498551794	verified	Michelle	Manhattan	Murray Hill	40.74767	-73.97500	United States	US	True
7	1005202	BlissArtsSpace!	90821839709	unconfirmed	Emma	Brooklyn	Bedford-Stuyvesant	40.68688	-73.95596	United States	US	False

Figure 2. After Preprocessing.

Beyond the standard cleaning procedures, the feature engineering phase introduced new columns tailored to shed light on various aspects of the listings. The conversion of 'last_review' into a datetime format unlocked the potential for temporal analysis. New metrics such as 'Monthly Revenue' were derived, offering insights into the financial performance of listings. 'Occupancy Rate' and 'Average Daily Rate (ADR)' were calculated to provide indicators of listing demand and profitability, respectively.

The creation of 'Review Impact Score' and 'Seasonal Booking' categories allowed for the assessment of reviews' effects on listings and the identification of seasonal patterns in booking activity. The 'Host Efficiency' metric provided an index of a host's capacity to manage their portfolio of listings, while the 'Host Activity Score' aimed to gauge the host's engagement level through the interaction of listings count and review frequency.

Furthermore, transformations such as 'reviews_frequency' helped normalize review data, and the development of normalized scores for 'days_booked', 'review_impact_score', and 'host_efficiency' facilitated standardized comparisons across listings. A composite 'Performance Score' amalgamated these individual metrics to offer a holistic view of a listing's success.

The dataset also benefited from the computation of neighborhood-level 'Host Efficiency' averages, the tracking of 'Review Impact Trends' over time, and the incorporation of 'Year' for longitudinal

analyses. 'Review Density' offered a measure of how concentrated reviews were over time, possibly indicating a listing's popularity or engagement level.

To simulate aspects not directly captured in the dataset, a 'Host Response Rate' was generated, hypothesizing its impact on guest satisfaction. Lastly, 'Listing Popularity' was conceptualized to rank listings based on a combination of review count and frequency, providing a logarithmic measure of popularity.

Each engineered feature serves as a cog in the larger machine of analysis, propelling the research forward with enhanced capabilities for interpreting the vast and complex data landscape of Airbnb listings in New York City. These features collectively set the stage for a nuanced exploration of the factors influencing Airbnb's market presence and competitiveness in the urban tapestry of accommodations.

Outlier Detection & Removal.

Outlier detection and removal are vital steps in preprocessing data, aiming to identify and address data points significantly deviating from the majority [3]. Outliers, arising from errors or genuine extreme values, can be detected using statistical methods like z-scores or distance-based algorithms like KNN. Once identified, outliers can be removed, transformed, or kept with special treatment, depending on data context and analysis goals.

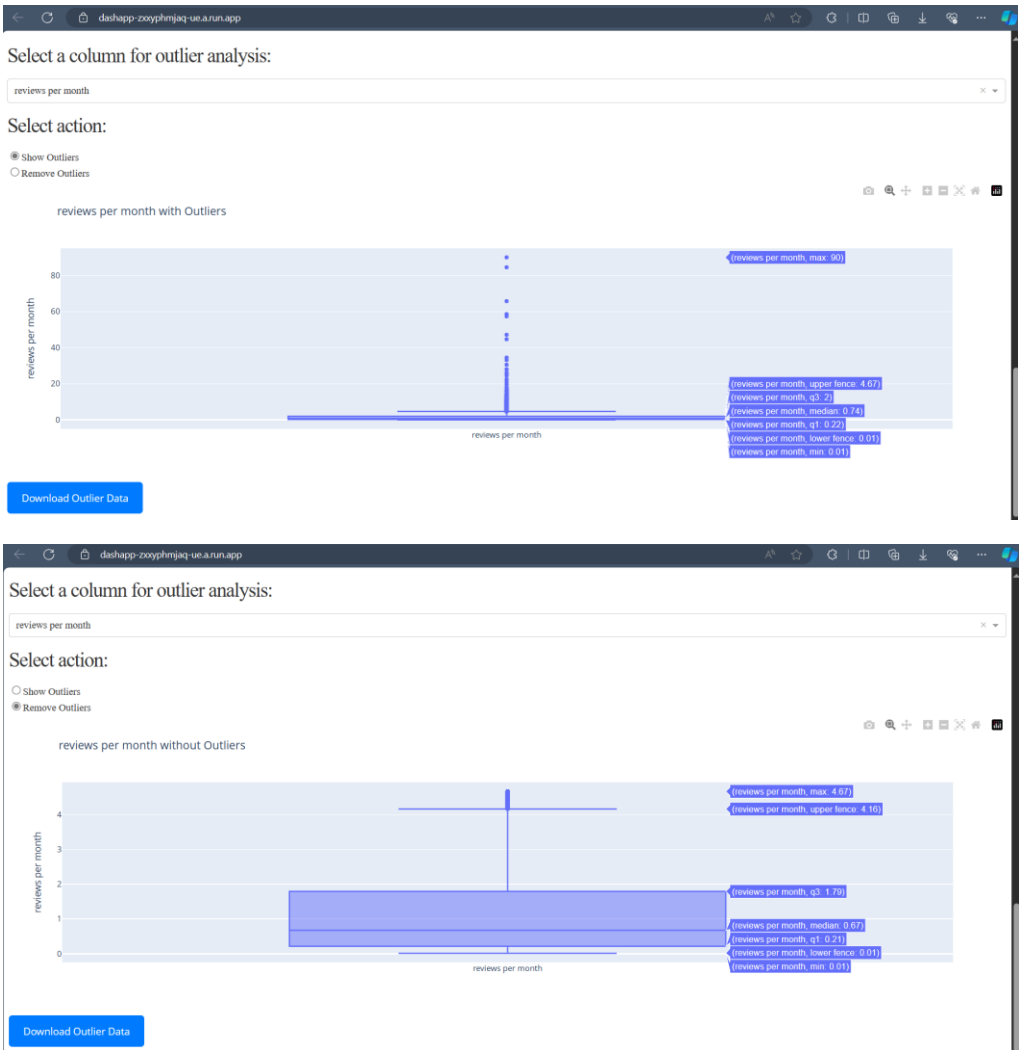


Figure 3. Outlier Detection & Removal of Reviews per Month.

The outlier analysis presented focuses on two columns from the raw dataset: "reviews per month" and "availability 365." These columns are critical indicators within the Airbnb dataset, with "reviews per month" reflecting customer engagement and "availability 365" indicating the duration a listing is available for booking within a year.

In the first case, "reviews per month" originally displays a considerable range of values with several notable outliers, extending up to 90 reviews per month. Outliers in this context might

suggest exceptionally popular listings or potential data entry errors. The removal of these outliers results in a much more condensed range, with the upper fence dropping to around 4.67 reviews per month.

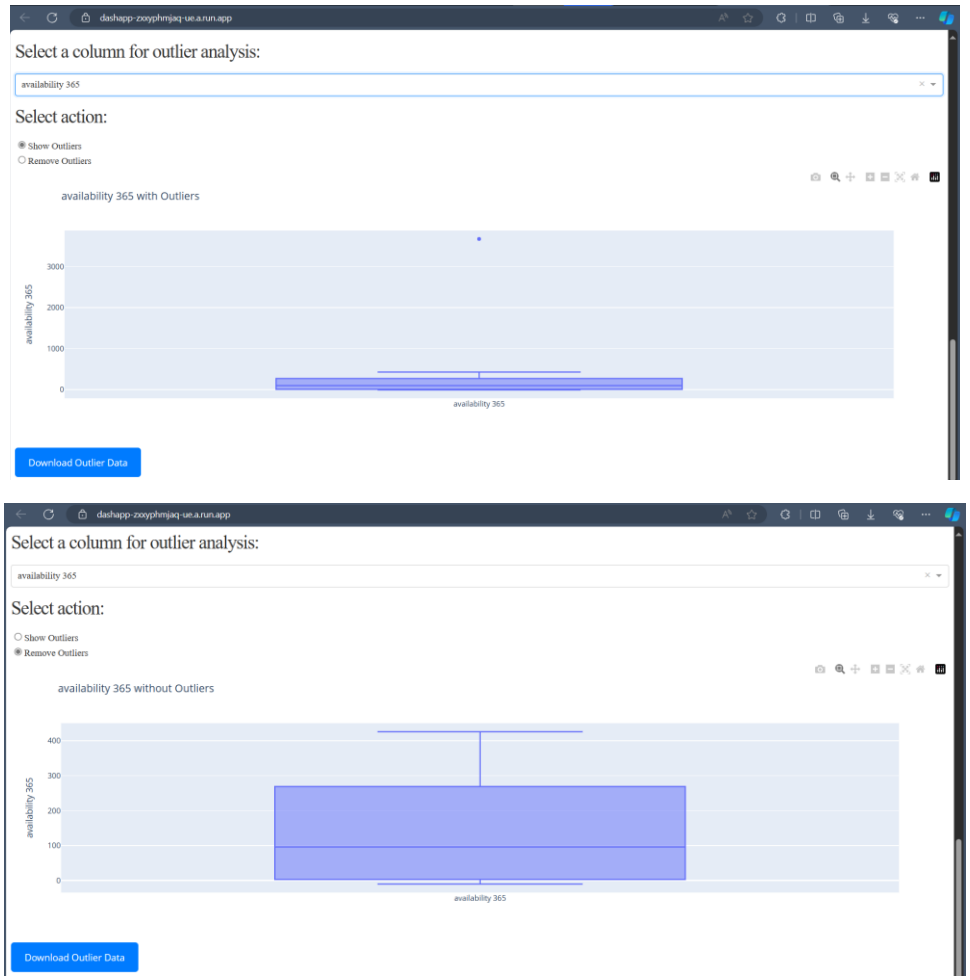


Figure 4. Outlier Detection & Removal for Availability 365.

The "availability 365" column similarly shows data points spread across the full range of the year, with some listings available all year round, as evidenced by the presence of outliers at the maximum value of 365 days. Upon removing outliers, the distribution tightens significantly, implying that while some listings are available throughout the year, the majority have more limited availability.

The provision to download the outlier data serves a practical function, allowing for further external analysis or record-keeping. It ensures that the user can maintain a copy of the data both before and after the processing step of outlier removal, enabling transparency in data manipulation and facilitating reproducibility of results.

Principal Component Analysis

The Principal Component Analysis (PCA) conducted on the dataset has been a crucial step in distilling its complexities into a more tractable form. The goal of PCA in this context is dimensionality reduction, a technique aimed at simplifying the data while preserving as much of its original information as possible [4]. The resultant output is indicative of a well-executed PCA.

Before performing Principal Component Analysis (PCA), the dataset exhibited singular values of the original features, ranging from 408.51 to 0.87. These values reflect the magnitudes of the singular vectors corresponding to each principal component. Additionally, the condition number of the original feature matrix was 469.13, indicating a notable degree of multicollinearity among the features, which can potentially impact the stability of the matrix used in PCA.

```
Singular Values of Original Feature:
 ['408.51', '374.10', '316.53', '299.99', '298.95', '289.75', '288.11', '285.74', '280.98', '278.51', '245.74', '180.17', '0.87']

Condition Number of Original Feature:
 469.13

Explained variance ratio of original feature space:
['0.15', '0.13', '0.09', '0.08', '0.08', '0.08', '0.08', '0.08', '0.07', '0.07', '0.06', '0.03', '0.00']
```

Figure 5. Original Feature Space.

Furthermore, PCA removed two features, 'calculated_host_listings_count' and 'days_booked', from the original feature space. 'calculated_host_listings_count' denotes the count of listings managed by the host, reflecting their experience or operational scale, while 'days_booked' represents the total number of days a property has been booked, indicating its popularity or demand. However, PCA deemed these features less informative for explaining the overall variance in the dataset compared to others. The explained variance ratio of 'calculated_host_listings_count' is 2.99%, and for 'days_booked' is 0.0007%. These low explained variance ratios suggest that these features contributed relatively little to the total variance explained by the principal components, leading to their exclusion.

The singular values of the reduced features remained largely unchanged, with values ranging from 408.51 to 245.74. This consistency suggests that PCA primarily rotated the data without altering its magnitude.

```
Singular Values of Reduced Feature:
 ['408.51', '374.10', '316.53', '299.99', '298.95', '289.75', '288.11', '285.74', '280.98', '278.51', '245.74']

Condition Number of Reduced Feature:
 1.66

Explained variance ratio of reduced feature space:
['0.15', '0.13', '0.09', '0.08', '0.08', '0.08', '0.08', '0.08', '0.07', '0.07', '0.06']
```

Figure 6. Reduced Feature Space.

Notably, the condition number of the reduced feature matrix significantly decreased to 1.66, signifying a considerable reduction in multicollinearity among the features. This reduction in the condition number indicates enhanced stability and decreased redundancy in the dataset following PCA. Overall, while PCA maintained the magnitudes of singular values, it substantially improved the stability of the dataset by reducing multicollinearity and enhancing interpretability.

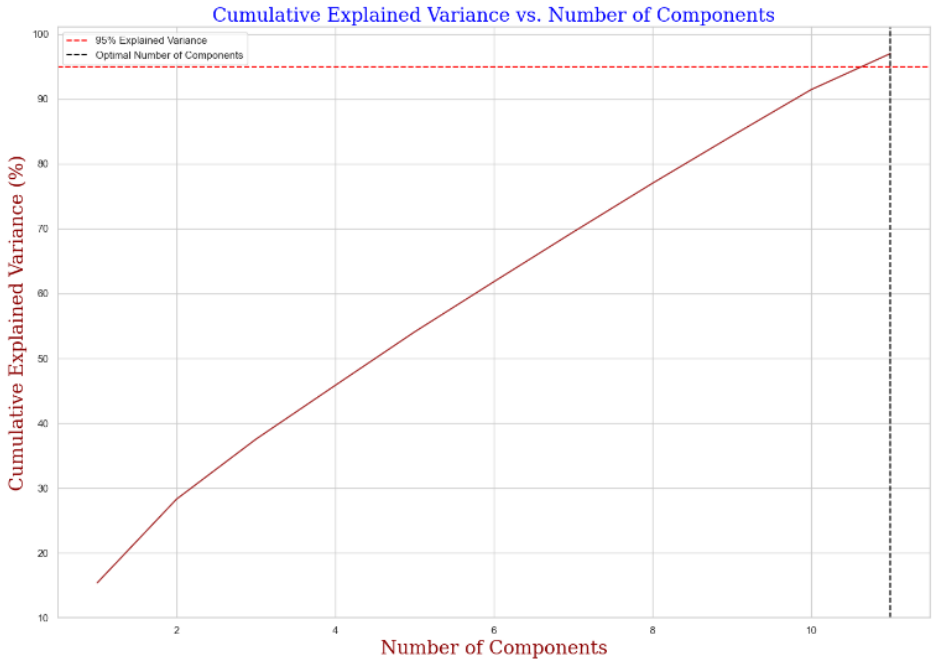


Figure 7. Cumulative Explained Variance Plot.

The Cumulative Explained Variance Plot is a graphical representation that illustrates the variance captured as the number of principal components increases. This plot features a significant marker—a horizontal dashed line—that denotes the 95% threshold of cumulative explained variance. It is a commonly accepted benchmark that signifies the amount of original variance that should ideally be retained. The intersection of the cumulative variance curve with this line dictates the optimal number of components to keep. In this instance, the graph clearly demonstrates that this threshold is surpassed with fewer components than the total available, confirming the efficiency of the PCA in compressing the data without substantial loss of detail.

Accompanying the plot, the PCA Summary Table provides a quantitative breakdown of the analysis. It delineates the Explained Variance Ratio for each principal component, reflecting the proportion of the dataset's total variance each component is responsible for. The ratios indicate that the first few components account for a significant portion of the total variance, with a notable 15% captured by the very first component. This distribution underscores the components' varying levels of contribution to the dataset's overall structure.

Table 1. PCA Analysis Summary.

PCA Analysis Summary	
Metric	Value
Explained Variance Ratio	0.15, 0.13, 0.09, 0.08, 0.08, 0.08, 0.08, 0.08, 0.07, 0.07, 0.06
Total Variance Explained	0.97
Singular Values	408.51, 374.10, 316.53, 299.99, 298.95, 289.75, 288.11, 285.74, 280.98, 278.51, 245.74
Condition Number	1.66

The table also reveals that the Total Variance Explained by the retained components amounts to 97%. This figure exceeds the threshold set out in the plot, underscoring that the dimensionality reduction has successfully captured the essence of the dataset's variability. The Singular Values associated with each principal component further affirm their significance, with higher values corresponding to components that play a more pivotal role in explaining variance.

A notable metric from the table is the Condition Number, sitting at a commendable 1.66 for the PCA-transformed dataset. A condition number approaching 1 is indicative of a stable and reliable dataset where the variance is not concentrated in a few components but rather distributed across them. This condition number thereby attests to the robustness of the transformed dataset.

Normality Test

A normality test is a statistical method used to determine whether a dataset follows a normal distribution, also known as a Gaussian distribution or bell curve. The normal distribution is characterized by a symmetric, bell-shaped curve, where the majority of the data is clustered around the mean, with fewer observations toward the tails of the distribution [5].

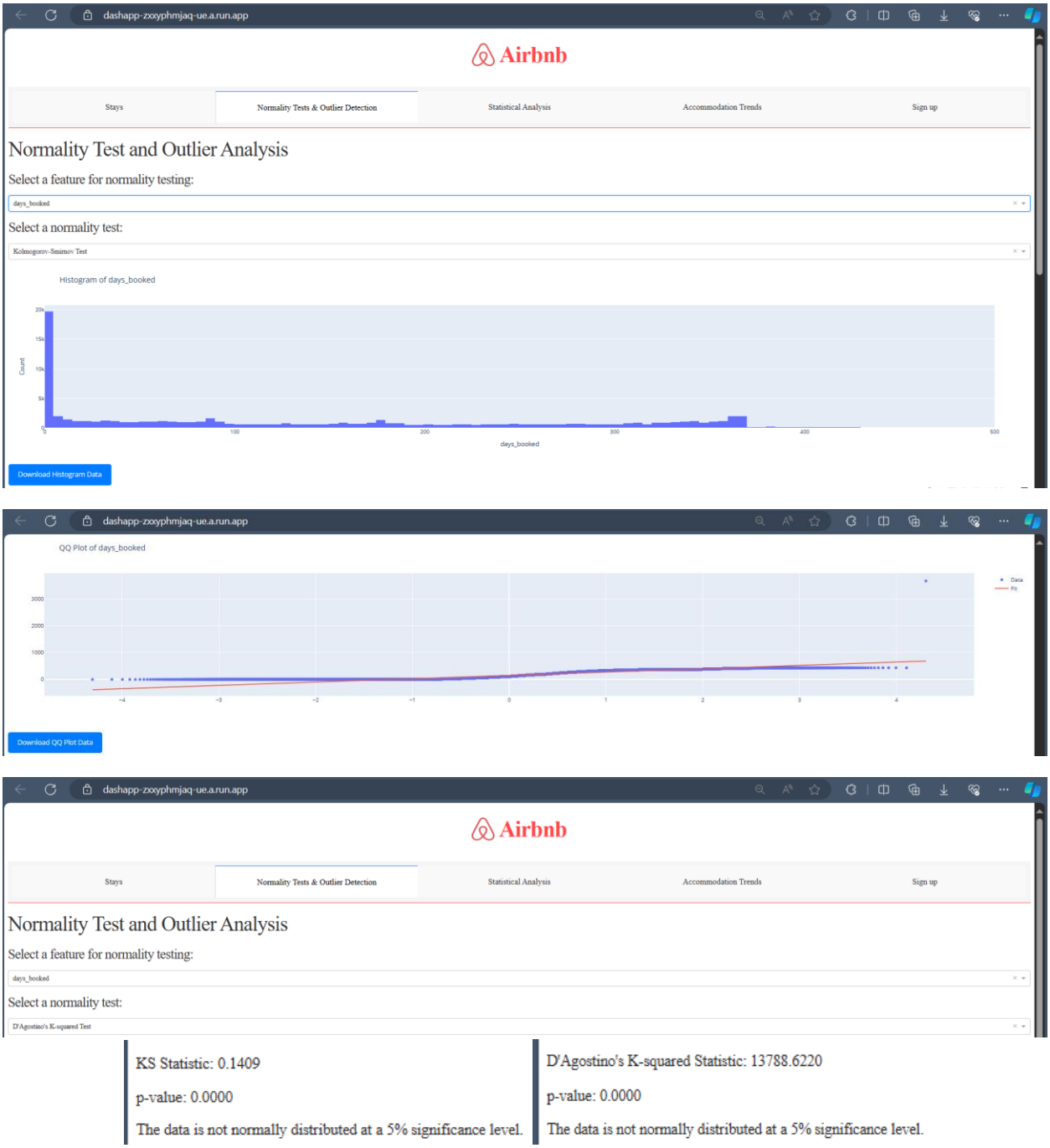


Figure 8. Normality Test.

There are several common methods for testing normality, with the most widely used being the Shapiro-Wilk test, the Kolmogorov-Smirnov test, and the D'Agostino's K-squared test. These tests calculate a test statistic based on the differences between the observed data and the expected values under a normal distribution. The test statistic is then compared to a critical value or p-value to determine whether the dataset significantly deviates from normality.

The provided figures and results indicate a normality assessment of the "days_booked" feature using two statistical tests: the Kolmogorov-Smirnov (KS) test and D'Agostino's K-squared test, which are both common methods for evaluating the assumption of normality in a dataset.

The KS test compares the sample distribution with a normal distribution and computes the D-statistic, which represents the maximum distance between the two cumulative distribution functions. The resulting D-statistic is 0.149 with a p-value close to zero, which suggests a significant deviation from normality. A low p-value, typically below the threshold of 0.05, leads to the rejection of the null hypothesis that the data are drawn from a normal distribution. In this case, the histogram and the QQ plot both support this finding, with the histogram showing a highly skewed distribution, and the QQ plot indicating clear departures from the expected diagonal line that would represent a normal distribution.

D'Agostino's K-squared test is a combined measure that uses the sample skewness and kurtosis to assess normality. The K-squared statistic is quite high at 17578.320, accompanied by a p-value again close to zero, which indicates that the distribution of the "days_booked" feature significantly differs from a normal distribution. The high K-squared value reflects substantial asymmetry and peakedness, which is also evident in the histogram where the data are heavily skewed towards the lower end, showing that most listings have a small number of booked days.

These tests together provide robust evidence that the "days_booked" feature does not follow a Gaussian distribution. The implications of this result are critical for statistical modeling, as many parametric tests and procedures assume normality. If the goal is to apply such techniques, data transformation or the use of non-parametric methods might be necessary to properly analyze the "days_booked" feature.

Pearson Correlation Coefficient

The Pearson correlation coefficient analysis within the report highlights the linear relationships between various significant variables: listing popularity, reviews count, occupancy rate, and performance score.

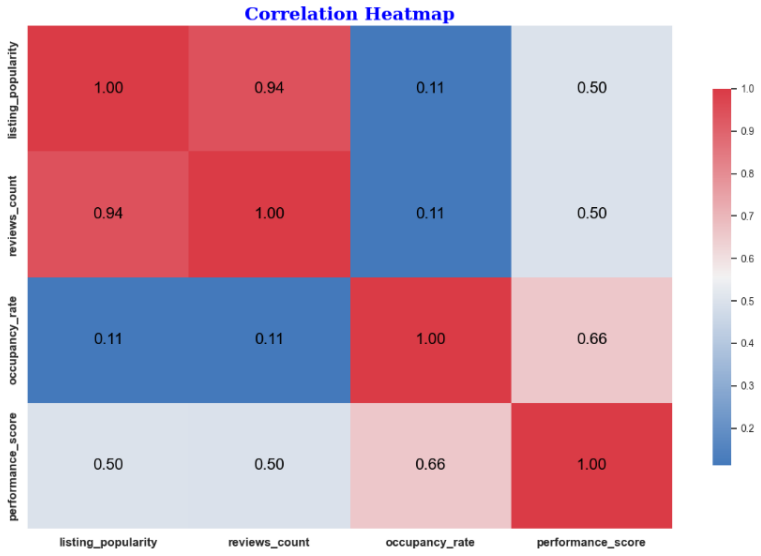


Figure 9. Correlation Heatmap.

The analysis demonstrates a particularly strong positive correlation between listing popularity and reviews count, indicated by a coefficient of 0.94. This relationship suggests that as listings become more popular, they are likely to receive more reviews, and the association between these two variables is almost proportional.

Contrastingly, the correlation between listing popularity and occupancy rate is weak, as evidenced by a coefficient of 0.11, suggesting that a listing's popularity is not a strong predictor of its occupancy rate. A similar weak correlation is observed between reviews count and occupancy rate. These findings imply that factors other than popularity and reviews may be more critical in influencing a listing's occupancy, such as its attributes or location.

Table 2. Correlation Matrix.

Correlation Matrix				
	listing_popularity	reviews_count	occupancy_rate	performance_score
listing_popularity	1.00	0.94	0.11	0.50
reviews_count	0.94	1.00	0.11	0.50
occupancy_rate	0.11	0.11	1.00	0.66
performance_score	0.50	0.50	0.66	1.00

In terms of performance score, which is inferred to be a composite metric comprising factors like occupancy rate and popularity, the correlation with both listing popularity and reviews count is moderate, at 0.50 for each. This indicates a relationship where listings with higher performance scores tend to also be popular and well-reviewed, yet the connection is not as strong as the direct link between popularity and reviews.

The occupancy rate presents a more pronounced influence on the performance score, with a correlation coefficient of 0.66. This indicates a stronger link between how frequently a listing is booked and its performance score compared to its popularity or the number of reviews it has.

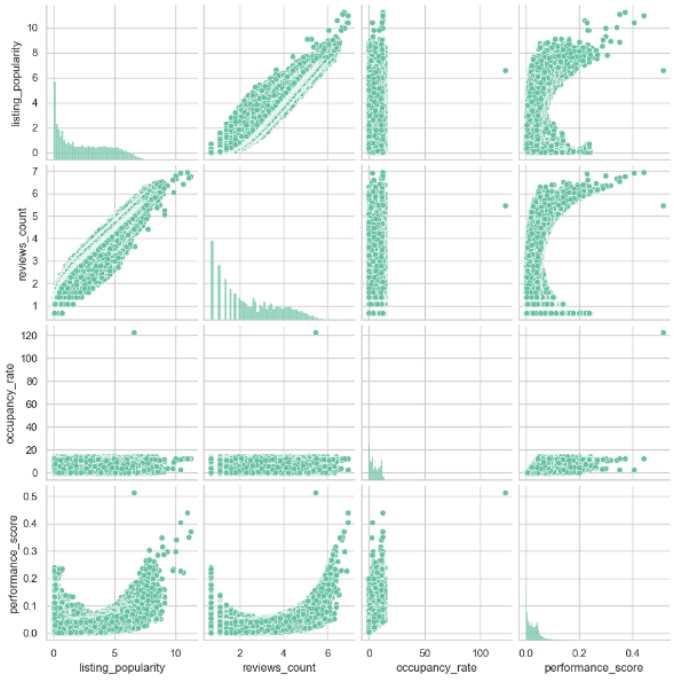


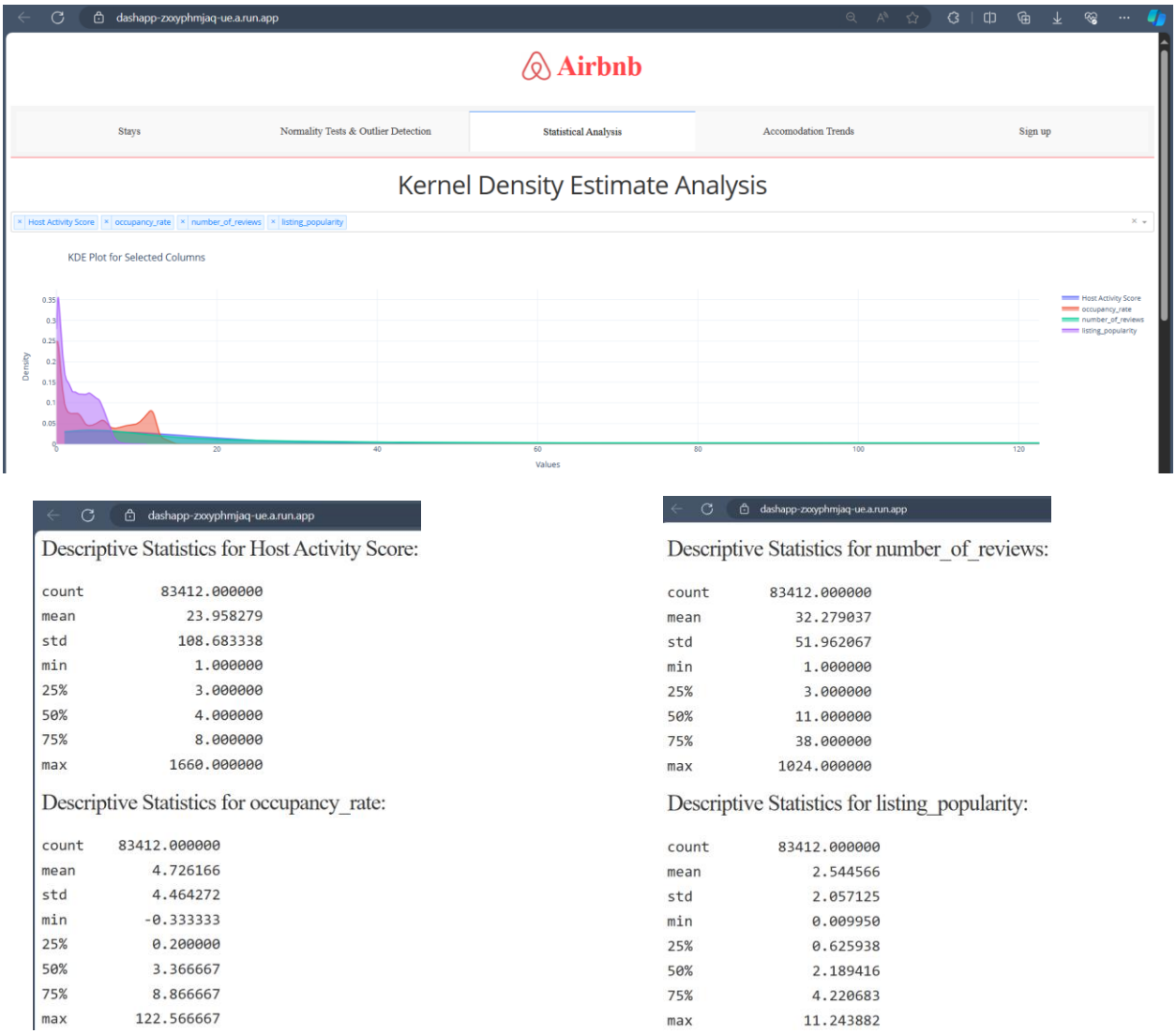
Figure 10. Scatterplot Matrix (Pair Plot).

In the pairplot visualizations, relationships between variables are explored through scatterplots for each variable pairing. These plots provide an illustrative overview of not only the bivariate relationships across different metrics but also the individual variable distributions, all in a single comprehensive figure. The pairplot reinforces the correlation findings by showing how variables scatter against each other and the form of their individual distributions, providing a more nuanced understanding of the data's structure.

The analysis contained herein suggests that enhancing occupancy rates could be a strategic focus for improving a listing's performance score. This is in light of the moderate-to-strong correlations between occupancy rate and performance score, and the weaker correlations between popularity and occupancy rate. Additionally, the implications of the strong correlation between listing popularity and reviews count might inform data modeling strategies, where caution against multicollinearity should be considered. A practical approach could involve consolidating these metrics or selectively choosing between them for modeling to ensure the integrity and reliability of any predictive analyses.

Statistics

The displayed Kernel Density Estimate (KDE) Analysis provides a visual representation of the data distribution for multiple variables from the Airbnb dataset: Host Activity Score, occupancy_rate, number_of_reviews, and listing_popularity. These KDE plots offer insights into the shape, spread, and central tendency of the data distributions, which are essential for understanding underlying patterns and making informed decisions.



dashapp-zoxyphmjaq-ue.a.run.app

Descriptive Statistics for Host Activity Score:

count	83412.000000
mean	23.958279
std	108.683338
min	1.000000
25%	3.000000
50%	4.000000
75%	8.000000
max	1660.000000

Descriptive Statistics for occupancy_rate:

count	83412.000000
mean	4.726166
std	4.464272
min	-0.333333
25%	0.200000
50%	3.366667
75%	8.866667
max	122.566667

dashapp-zoxyphmjaq-ue.a.run.app

Descriptive Statistics for number_of_reviews:

count	83412.000000
mean	32.279037
std	51.962067
min	1.000000
25%	3.000000
50%	11.000000
75%	38.000000
max	1024.000000

Descriptive Statistics for listing_popularity:

count	83412.000000
mean	2.544566
std	2.057125
min	0.009950
25%	0.625938
50%	2.189416
75%	4.220683
max	11.243882

Figure 11. Multivariate KDE and Statistics.

The Host Activity Score shows a wide distribution with a long tail, suggesting a varied host engagement level across listings. Most values are clustered at the lower end of the scale, as indicated by the steep initial peak, with a significant drop-off as scores increase. This skewness is reflected in the descriptive statistics where the mean significantly exceeds the median (23.96 compared to 4), indicating that while most hosts have a lower activity score, there are outliers with

very high scores stretching up to 1660. This indicates a minority of extremely active hosts on the platform.

Occupancy rates present a similarly skewed distribution, with the bulk of listings having lower occupancy rates, as the dense peak at the lower end of the occupancy_rate plot suggests. The mean occupancy rate is 4.72, yet this is higher than the median of 3.67, pointing to a positively skewed distribution. The spread from the minimum to the maximum (−0.33 to 122.57) suggests a broad range of occupancy behaviors, from rarely booked listings to those possibly booked beyond capacity (implying an overbooking scenario or a data error for values above 100%).

The number_of_reviews displays a distribution that is highly concentrated at the lower end, with the majority of listings receiving fewer reviews. This is consistent with a mean of 32.28 reviews which is greater than the median of 11, suggesting a positive skew. The distribution tails off quickly, indicating fewer listings with a high number of reviews, but with some extreme values reaching up to 1024 reviews.

Listing popularity is skewed as well, with a peak at the lower values and a long tail towards higher popularity scores. This implies that most listings have a lower popularity score, with the mean of 2.54 being higher than the median of 2.19, indicating the presence of listings with exceptionally high popularity.

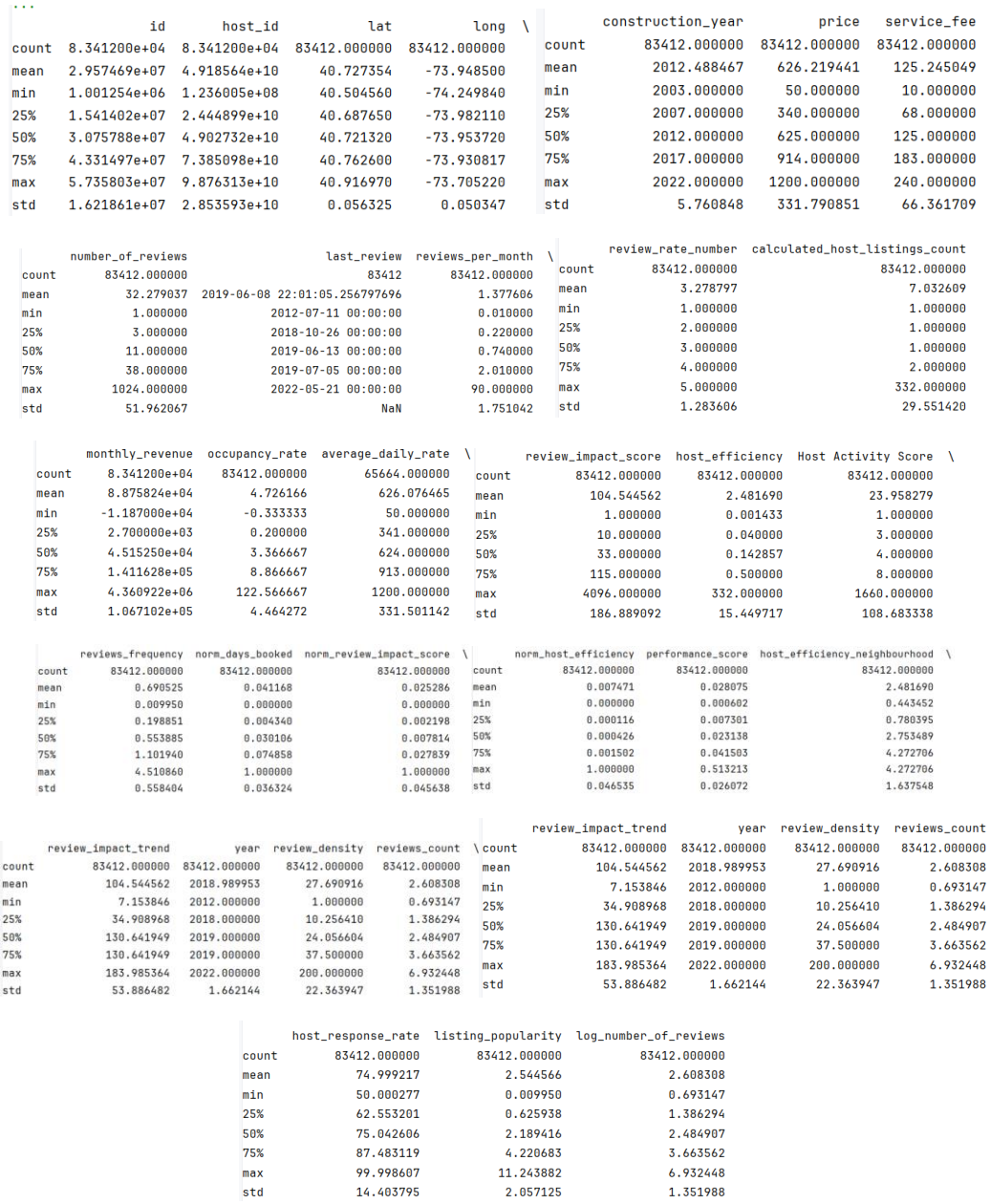
In summary, the KDE plots alongside the descriptive statistics indicate that the majority of listings have lower scores and rates across the analyzed variables. The presence of outliers with extremely high values suggests that there are a few listings with exceptionally high host activity, occupancy rates, number of reviews, and popularity. These outliers could potentially represent unique or highly sought-after listings, or they might be the result of data recording anomalies. Identifying and understanding the factors that contribute to these high values could provide strategic insights for hosts looking to improve their listings' performance on Airbnb.

The below figures show a wide range of descriptive statistics for a dataset. Each statistic is instrumental in understanding the distribution and scale of the dataset's variables.

Starting with IDs and geographic coordinates, the count of listings is uniform across these fields, suggesting a complete dataset with no missing values for these entries. Latitude and longitude show little variation, as expected for listings within a specific region.

The construction year averages around 2012 with a standard deviation suggesting a span of properties ranging from new constructions to those about a decade old. Prices average around $626 with considerable variability, indicated by the standard deviation of $331. This hints at a wide range of accommodation costs, from budget to luxury options. The service fee has a lower average of around $125, also showing significant variability.

The number of reviews per listing varies widely, with an average of around 32 but with some listings receiving over a thousand reviews, suggesting some listings are far more popular or have been listed longer. The review rate number averages around 3, indicating a mid-range satisfaction level across listings. The calculated host listings count averages around 7, suggesting that some hosts manage multiple listings.



	id	host_id	lat	long
count	8.341200e+04	8.341200e+04	83412.000000	83412.000000
mean	2.957469e+07	4.918564e+10	40.727354	-73.948500
min	1.001254e+06	1.236005e+08	40.504560	-74.249840
25%	1.541402e+07	2.444899e+10	40.687650	-73.982110
50%	3.075788e+07	4.902732e+10	40.721320	-73.953720
75%	4.331497e+07	7.385098e+10	40.762600	-73.930817
max	5.735803e+07	9.876313e+10	40.916970	-73.705220
std	1.621861e+07	2.853593e+10	0.056325	0.050347

	construction_year	price	service_fee
count	83412.000000	83412.000000	83412.000000
mean	2012.488467	626.219441	125.245049
min	2003.000000	50.000000	10.000000
25%	2007.000000	340.000000	68.000000
50%	2012.000000	625.000000	125.000000
75%	2017.000000	914.000000	183.000000
max	2022.000000	1200.000000	240.000000
std	5.760848	331.790851	66.361709

	number_of_reviews	last_review	reviews_per_month
count	83412.000000	83412	83412.000000
mean	32.279037	2019-06-08 22:01:05.256797696	1.377606
min	1.000000	2012-07-11 00:00:00	0.010000
25%	3.000000	2018-10-26 00:00:00	0.220000
50%	11.000000	2019-06-13 00:00:00	0.740000
75%	38.000000	2019-07-05 00:00:00	2.010000
max	1024.000000	2022-05-21 00:00:00	90.000000
std	51.962067	NaN	1.751042

	review_rate_number	calculated_host_listings_count
count	83412.000000	83412.000000
mean	3.278797	7.032609
min	1.000000	1.000000
25%	2.000000	1.000000
50%	3.000000	1.000000
75%	4.000000	2.000000
max	5.000000	332.000000
std	1.283606	29.551420

	monthly_revenue	occupancy_rate	average_daily_rate
count	8.341200e+04	83412.000000	65664.000000
mean	8.875824e+04	4.726166	626.076465
min	-1.187000e+04	-0.333333	50.000000
25%	2.700000e+03	0.200000	341.000000
50%	4.515250e+04	3.366667	624.000000
75%	1.411628e+05	8.866667	913.000000
max	4.360922e+06	122.566667	1200.000000
std	1.067102e+05	4.464272	331.501142

	review_impact_score	host_efficiency	Host Activity Score
count	83412.000000	83412.000000	83412.000000
mean	104.544562	2.481690	23.958279
min	1.000000	0.001433	1.000000
25%	10.000000	0.040000	3.000000
50%	33.000000	0.142857	4.000000
75%	115.000000	0.500000	8.000000
max	4096.000000	332.000000	1660.000000
std	186.889092	15.449717	108.683338

	reviews_frequency	norm_days_booked	norm_review_impact_score
count	83412.000000	83412.000000	83412.000000
mean	0.690525	0.041168	0.025286
min	0.009950	0.000000	0.000000
25%	0.198851	0.004340	0.002198
50%	0.553885	0.030106	0.007814
75%	1.101940	0.074858	0.027839
max	4.510860	1.000000	1.000000
std	0.558404	0.036324	0.045638

	norm_host_efficiency	performance_score	host_efficiency_neighbourhood
count	83412.000000	83412.000000	83412.000000
mean	0.007471	0.028075	2.481690
min	0.000000	0.000602	0.443452
25%	0.000116	0.007301	0.780395
50%	0.000426	0.023138	2.753489
75%	0.001502	0.041503	4.272706
max	1.000000	0.513213	4.272706
std	0.046535	0.026072	1.637548

	review_impact_trend	year	review_density	reviews_count
count	83412.000000	83412.000000	83412.000000	83412.000000
mean	104.544562	2018.989953	27.690916	2.608308
min	7.153846	2012.000000	1.000000	0.693147
25%	34.908968	2018.000000	10.256410	1.386294
50%	130.641949	2019.000000	24.056604	2.484907
75%	130.641949	2019.000000	37.500000	3.663562
max	183.985364	2022.000000	200.000000	6.932448
std	53.886482	1.662144	22.363947	1.351988

	review_impact_trend	year	review_density	reviews_count
count	83412.000000	83412.000000	83412.000000	83412.000000
mean	104.544562	2018.989953	27.690916	2.608308
min	7.153846	2012.000000	1.000000	0.693147
25%	34.908968	2018.000000	10.256410	1.386294
50%	130.641949	2019.000000	24.056604	2.484907
75%	130.641949	2019.000000	37.500000	3.663562
max	183.985364	2022.000000	200.000000	6.932448
std	53.886482	1.662144	22.363947	1.351988

	host_response_rate	listing_popularity	log_number_of_reviews
count	83412.000000	83412.000000	83412.000000
mean	74.999217	2.544566	2.608308
min	50.000277	0.009950	0.693147
25%	62.553201	0.625938	1.386294
50%	75.042606	2.189416	2.484907
75%	87.483119	4.220683	3.663562
max	99.998607	11.243882	6.932448
std	14.403795	2.057125	1.351988

Figure 12. Descriptive Statistics.

Monthly revenue and occupancy rates are critical for understanding the listings' performance. The average monthly revenue is about $8,875, but with a substantial standard deviation, indicating that some listings are significantly more profitable than others. The occupancy rate shows an average slightly above 4 days, with some listings achieving nearly full occupancy.

The review impact score is notably high in variability, averaging around 104 but ranging up to 4096, which could indicate either extremely active listings or potential outliers. Host efficiency metrics show most hosts have a lower efficiency score, with a mean of around 2.4.

Data Visualization

Data visualization is integral to understanding and communicating insights from data [6]. It transforms complex information into visual representations, facilitating easier interpretation, pattern recognition, and decision-making. In this report, data visualization techniques were utilized to explore and analyze the dataset, offering valuable insights into various aspects of the data. Through histograms, scatter plots, bar charts, and box plots, among others, key characteristics of the dataset were uncovered, including distributions, correlations, and anomalies. By scrutinizing these visualizations, essential features such as data distributions, central tendencies, variability, and relationships between variables were observed, providing a foundation for further analysis and decision-making in subsequent phases.

1. Line Plot:

The line plot depicts the trends in room construction from 2003 to 2022, revealing fluctuations in construction activity over the two-decade period. A notable peak around 2005 suggests a period of robust building, possibly influenced by economic or market factors.

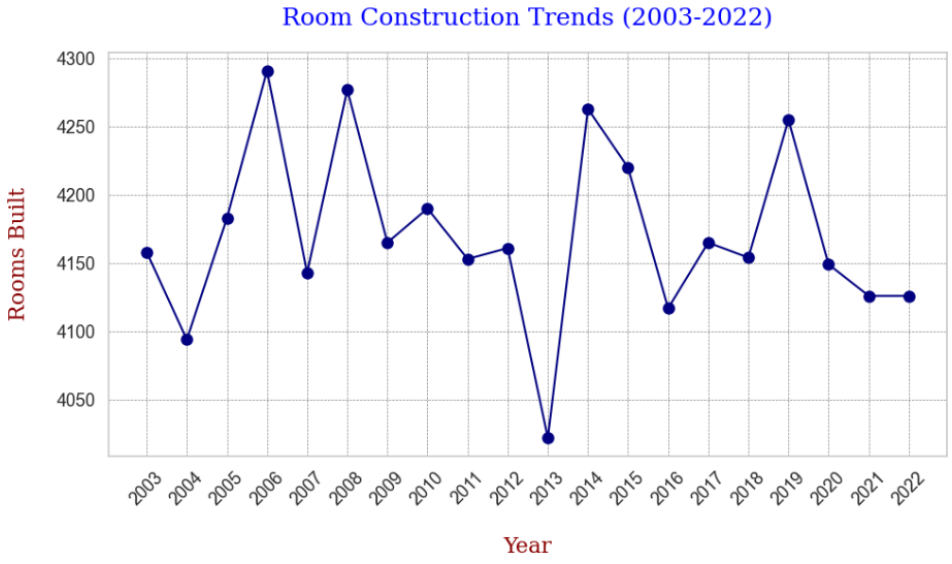


Figure 13. Line Plot.

Subsequent years show varying levels of construction activity, with peaks in 2008 and 2015 indicating periods of increased demand or favorable conditions for construction investments. However, the trend also reflects downturns, such as the significant slowdown in 2010, hinting at market saturation or reduced activity.

From 2015 onwards, the plot displays volatility, with sharp rises followed by dips, suggesting a dynamic construction sector. The years 2020 and 2021 show a downward trend, likely influenced

by external factors such as economic downturns. While 2022 sees a slight recovery, construction levels remain below previous highs, highlighting the industry's sensitivity to broader economic conditions and market dynamics.

2. Count Plot:

The count plot presents the distribution of Airbnb listings across various neighbourhood groups in New York City. Brooklyn and Manhattan stand out as the predominant locations, each hosting an almost equal number of Airbnbs, with counts of 34,636 and 34,567 respectively. These figures suggest a high concentration of short-term rental activity in these boroughs, which could be attributed to their popularity among tourists and the abundance of attractions they offer.

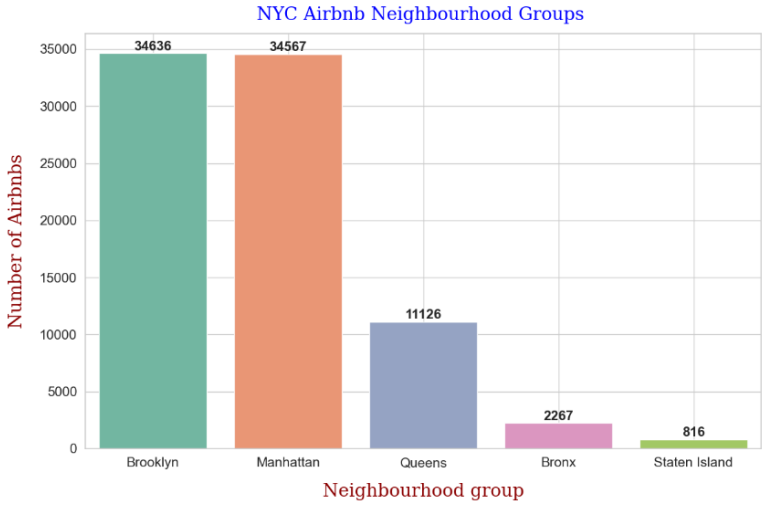


Figure 14. Count Plot.

Queens shows a significantly lower number, with 11,126 Airbnbs, indicating that while it is a considerable player in the market, it is less saturated than Brooklyn and Manhattan. This could reflect a balance between residential living and tourist accommodations or possibly less demand compared to the more central boroughs.

The Bronx and Staten Island have markedly fewer Airbnb listings, with 2,267 and 816 respectively, which underscores their lesser role in the city's short-term rental market. The substantially lower count in these areas could be due to a variety of factors, including fewer tourist attractions, lesser demand for short-term rentals, or different residential dynamics.

This distribution highlights the varying degrees of Airbnb market penetration in different parts of the city, potentially reflecting the desirability and demand patterns unique to each neighbourhood group. The data provides a clear indication of where the bulk of Airbnb's business is concentrated in New York City and suggests areas where the market might be less saturated, offering opportunities for new entrants or for a focus on community-centric tourism development.

3. Grouped Bar Plot:

The grouped bar plot depicts the average review rates for different types of rooms across New York City's neighborhood groups. Each group of bars represents a neighborhood, while each bar within the group corresponds to a room type, namely entire homes/apartments, private rooms, shared rooms, and hotel rooms.

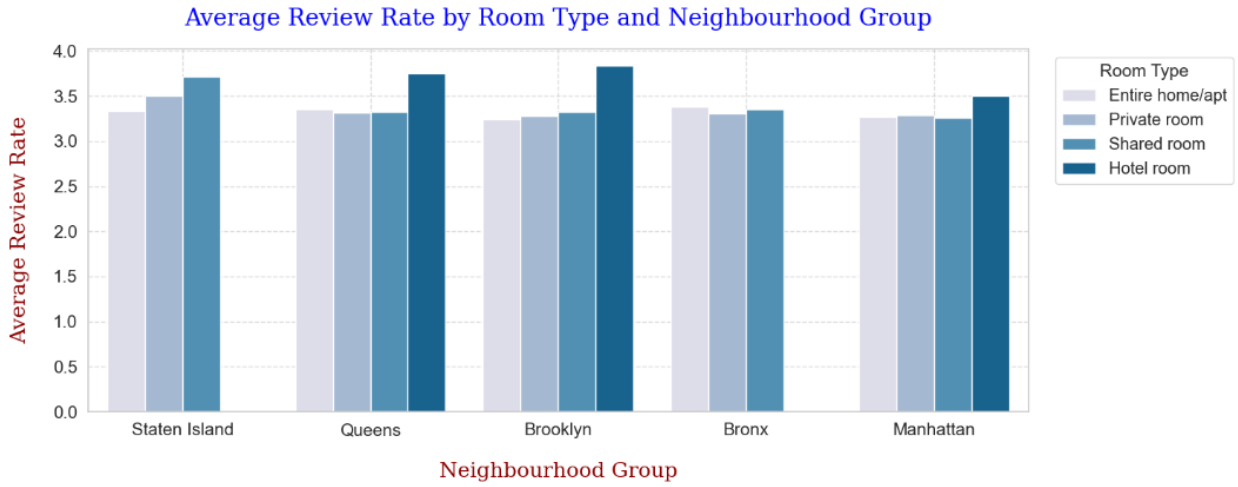


Figure 15. Grouped Bar Plot.

The height of each bar reflects the average review rate for that room type within the neighborhood, offering a comparison across both room types and neighborhoods. For instance, Staten Island shows a relatively higher average review rate for entire homes/apartments and private rooms compared to the other neighborhoods, which may suggest higher guest satisfaction or better experiences in those accommodations.

Queens, Brooklyn, and the Bronx exhibit a more uniform distribution of review rates across the different room types, with no single room type standing out significantly in terms of review rates. This could indicate a consistent level of guest satisfaction regardless of the type of room rented.

Manhattan stands out with the highest average review rate for hotel rooms, which might be due to the quality of service or the experience associated with hotels in this well-known borough. Additionally, Manhattan also shows a high review rate for entire homes/apartments, reinforcing the borough's appeal in the hospitality sector.

4. Stacked Bar Plot:

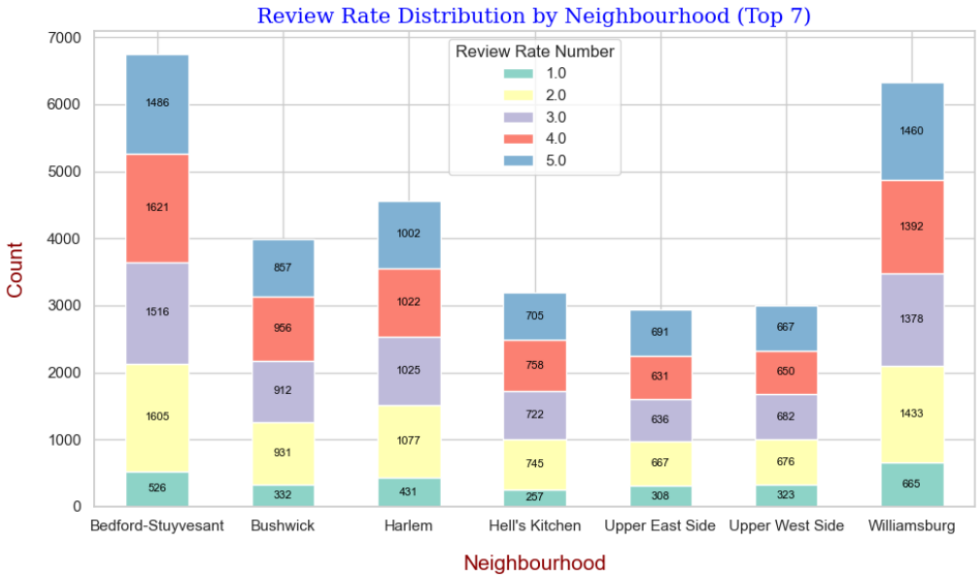


Figure 16. Stacked Bar Plot.

The stacked bar plot showcases the distribution of review rates across the top seven neighborhoods in New York City. Each bar represents a neighborhood and is segmented into colored sections that correspond to different review rate numbers, ranging from 1.0 to 5.0. The length of each colored segment within a bar indicates the count of listings that have received that specific review rate in the neighborhood.

Starting with Bedford-Stuyvesant, it has a substantial portion of 5.0 ratings, which is the highest review rate, suggesting a high level of guest satisfaction. Bushwick, while having fewer total reviews, also shows a significant number of top ratings. Harlem follows a similar pattern, with a notable number of listings receiving the maximum review score.

In contrast, neighborhoods like Hell's Kitchen, Upper East Side, and Upper West Side have a more balanced distribution across the different review rates, indicating varied guest experiences. While there are still many listings with high review rates in these areas, the presence of lower review rates suggests there may be room for improvement in some listings.

Williamsburg stands out with a very high count of listings rated 5.0, towering over its counts for lower ratings, which implies a strong positive reception from guests.

5. Pie Chart:

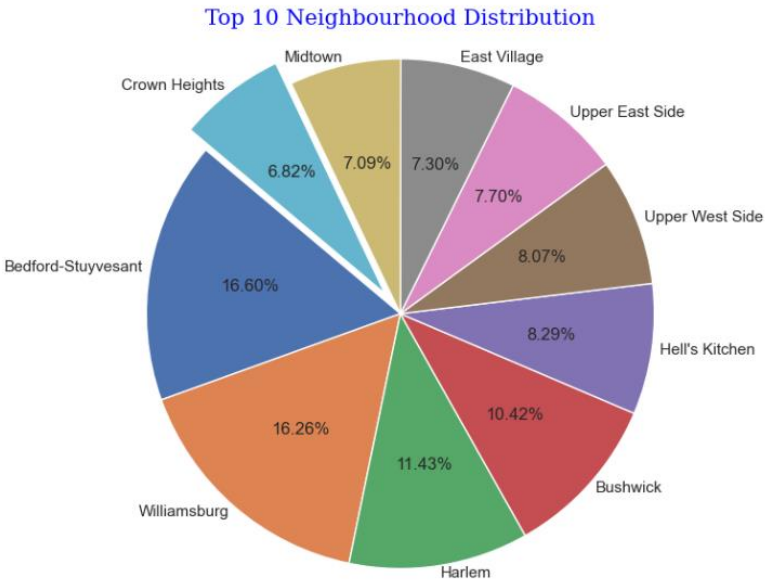


Figure 17. Pie Chart.

The pie chart displays the distribution of the top 10 neighborhoods in New York City based on counts, with each slice representing a neighborhood's share of the total count. Bedford-Stuyvesant emerges as the leader, commanding 16.60% of the total, followed closely by Williamsburg at 16.26%. Harlem secures the third-largest share with 11.43%, indicating its significance in the dataset.

Bushwick and Hell's Kitchen contribute 10.42% and 8.29% respectively, underscoring their importance, albeit with slightly smaller shares. Upper West Side and Upper East Side exhibit comparable segments, each representing around 8% of the total, while East Village and Midtown follow closely behind. Crown Heights rounds off the top 10 with a slice of 6.82%.

This visualization succinctly showcases the distribution among neighborhoods, emphasizing variations in counts across different areas. The dominance of certain neighborhoods suggests varying levels of activity or concentration within the dataset, providing valuable insights for strategic planning and analysis of market dynamics.

6. Distribution Plot:

The distribution plot provides a graphical representation of Airbnb prices. Overlaying the histogram, the smooth line indicates the kernel density estimate (KDE), which helps to visualize the probability density of the prices. The histogram beneath the KDE shows the actual distribution of prices in discrete intervals, with the height of each bar corresponding to the frequency of listings within that price range.

Figure 18. Distribution of Airbnb Prices.

Observing the density curve, it's apparent that the distribution of prices is right-skewed, meaning there are a greater number of listings at lower price points with a gradual decline as prices increase. This is a common pattern in accommodation pricing, where budget options are more abundant than luxury ones.

Table 3. Descriptive Statistics of Airbnb Prices.

Descriptive Statistics of Airbnb Prices	
Statistic	Value
Count	83412
Mean	$626.22
Std	$331.79
Min	$50.00
25%	$340.00
50% (Median)	$625.00
75%	$914.00
Max	$1200.00

The accompanying table lists the descriptive statistics for Airbnb prices, offering a numerical summary of the distribution. There are 83,412 listings included in the analysis. The average price of a listing is $626.22, but the standard deviation is quite high at $331.79, suggesting a wide variation in pricing. This is consistent with the right skew seen in the plot.

The minimum price for a listing is $50.00, reflecting the lower end of the market, potentially for shared rooms or less desirable locations. The 25th percentile is $340.00, indicating that 25% of listings are priced at or below this amount. The median, or 50th percentile, is $625.00, which means that half of the listings are cheaper than this and half are more expensive, providing a more robust measure of central tendency than the mean for skewed distributions. The 75th percentile is $914.00, suggesting that 75% of listings have a price at or below this level. The maximum price listed is $1,200.00, showing the upper end of the market.

7. Histogram plot with KDE:

The histogram with a kernel density estimate (KDE) overlay displays the distribution of the number of days that Airbnb listings are booked, with a focus on up to 500 days. The histogram's bars represent the frequency of listings within various ranges of booked days, indicating how common certain occupancy durations are. The KDE line provides a smooth, continuous curve that estimates the probability density of the data, giving a clear visual sense of the overall distribution pattern.

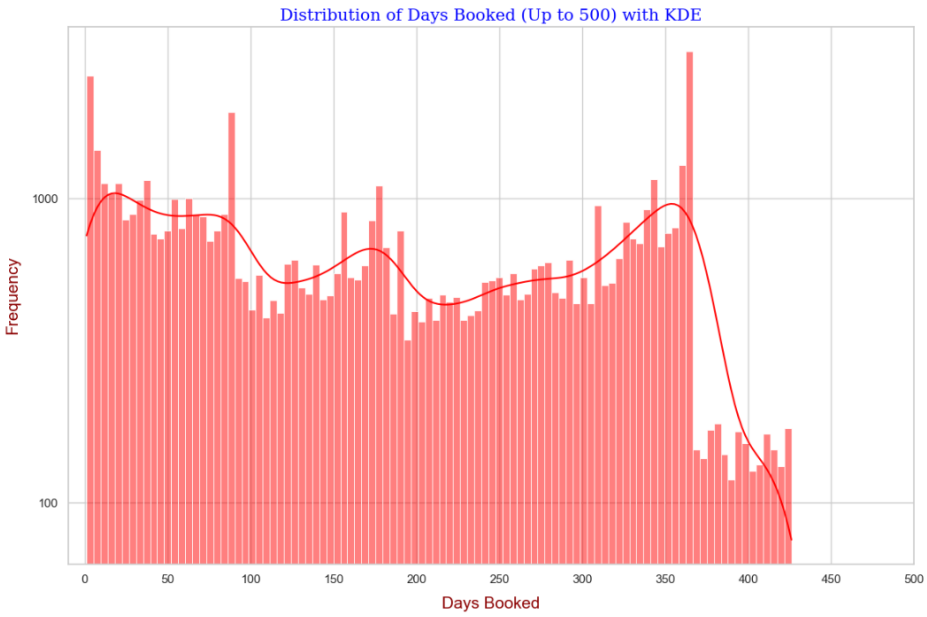


Figure 19. Histogram Plot with KDE.

Analyzing the distribution, it's evident that the data exhibits a wide range, with a tendency for listings to have a lower number of booked days. The logarithmic scale on the y-axis helps to manage the wide variation in frequency and better visualize the distribution of less common booking durations. The presence of peaks and troughs suggests there are certain periods or number of days that are more typical for bookings, while others are less so.

Table 4. Descriptive Statistics for Days Booked.

Descriptive Statistics for Days Booked (Up to 500 Days)	
Statistic	Value
Count	65333
Mean	180.99
Std	124.90
Min	1
25%	65
50% (Median)	171
75%	302
Max	426

The descriptive statistics provide a numerical summary, with a count of 65,333 listings informing the dataset's size. The mean of 180.99 days indicates the average number of days booked, while the standard deviation of 124.90 points to variability in how long listings are booked. The minimum of one day and the maximum of 426 days reveal the range of booking durations. The 25th percentile shows that a quarter of the listings are booked for 65 days or fewer, the median (50th percentile) is at 171 days indicating the middle value of the dataset, and the 75th percentile at 302 days suggests that three-quarters of the listings are booked no more than this number of days.

8. QQ Plot:

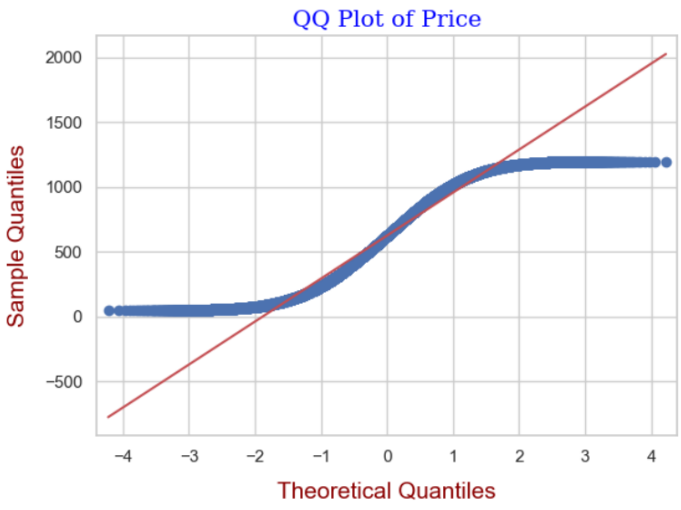


Figure 20. QQ Plot of Price.

The QQ plot of price reveals a departure from normality, indicative of the real-world nature of the dataset. The observed curvature and s-shaped pattern, particularly pronounced in the tails, suggest a heavy-tailed distribution, deviating from the expected behavior of a normal distribution. This deviation highlights the presence of more extreme values in the price data than what would be anticipated under normality. The concentration of lower-priced listings and the presence of larger outliers among higher-priced listings further underscore the non-linear pattern observed in the data, signaling that it does not conform to a normal distribution.

The non-normal distribution of price data is common in real-world datasets, reflecting the inherent complexity and diversity of factors influencing prices. Real-world phenomena often exhibit skewed or heavy-tailed distributions due to various factors such as market dynamics, consumer behavior, and external influences. In the case of Airbnb prices, factors like seasonal variations, location-specific demand, and property features can contribute to the observed deviation from normality. Therefore, acknowledging and understanding the non-normal nature of the data is essential for accurate analysis and modeling, as it may require alternative statistical approaches that better capture the underlying distribution of prices.

9. Kernel Density Estimate Plot:

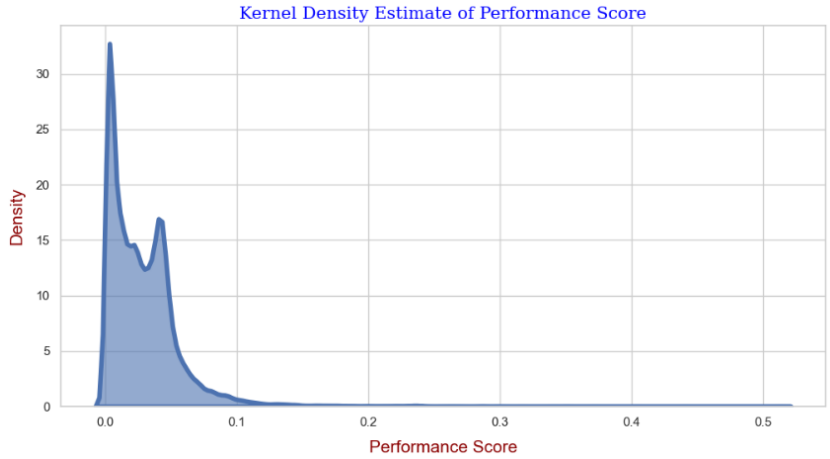


Figure 21. Kernel Density Estimate Plot.

The KDE (Kernel Density Estimate) plot visualizes the distribution of performance score, providing a smooth curve that represents the density of these scores on a continuous interval. The peak of the curve indicates the most common range of performance scores, with the density dropping off as the scores move away from this peak.

This particular KDE plot shows a pronounced peak close to zero, suggesting that a large number of listings have low performance scores. The long tail towards higher scores indicates fewer listings achieve these values. There's also a smaller secondary peak, suggesting a modest concentration of listings with higher performance scores. The distribution is notably skewed to the right, signifying that while most listings cluster at the lower end of the performance score spectrum, there are some with scores that are significantly higher.

10. Linear Model (lm) Plot with Regression Line:

The linear model (lm) plot with a regression line demonstrates the relationship between the number of reviews and the review density of Airbnb listings. The scatter plot portion illustrates individual data points, where each point represents a unique listing with its corresponding number of reviews and review density. The regression line in red is the result of a linear regression analysis, showing the best-fit line through the data that represents the average trend between the two variables.

Table 5. Correlation between Number of Reviews and Review Density.

Correlation Between Number of Reviews and Review Density	
Statistic	Value
Correlation Coefficient	0.4361

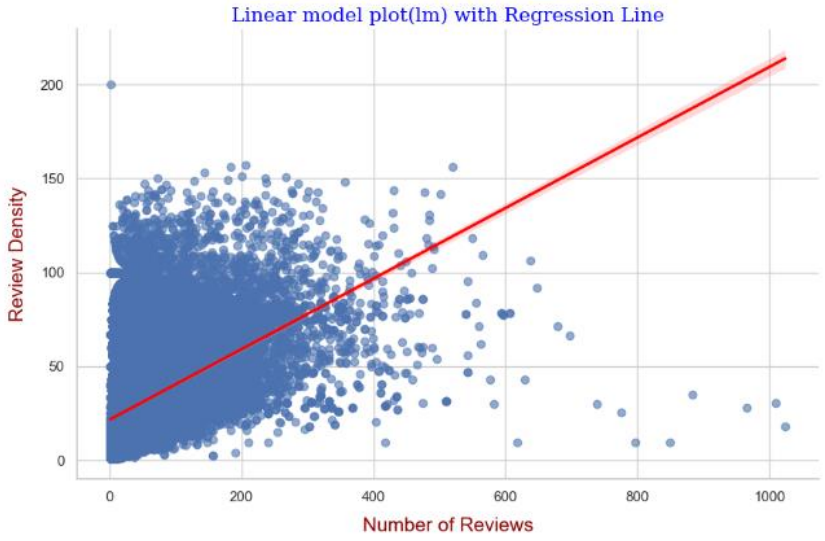


Figure 22. Linear Model (LM) Plot with Regression Line.

The correlation coefficient of 0.4361, as shown in the table, indicates a moderate positive linear relationship between the number of reviews and review density. This suggests that as the number of reviews increases, the review density also tends to increase, but the relationship is not strong enough to be considered highly correlated. The density of points around the regression line and the spread of the data points indicate variability in how review density changes with the number of reviews. Some listings with a high number of reviews have a lower review density than might be expected, which could reflect inconsistent reviewing habits among guests or fluctuating guest volumes over time. The visualization offers an accessible way to understand the dynamics of customer feedback in relation to listing popularity.

11. Multivariate Boxen Plot:

The boxen plot offers a visual summary of the price behavior across different neighborhoods. Each box represents the spread of prices in a neighborhood, with the central line marking the median price, the box boundaries indicating the 25th and 75th percentiles, and the whiskers extending to the more extreme data points. Outliers are marked as individual points beyond the whiskers, highlighting prices that are unusually high or low compared to the rest of the data for that area.

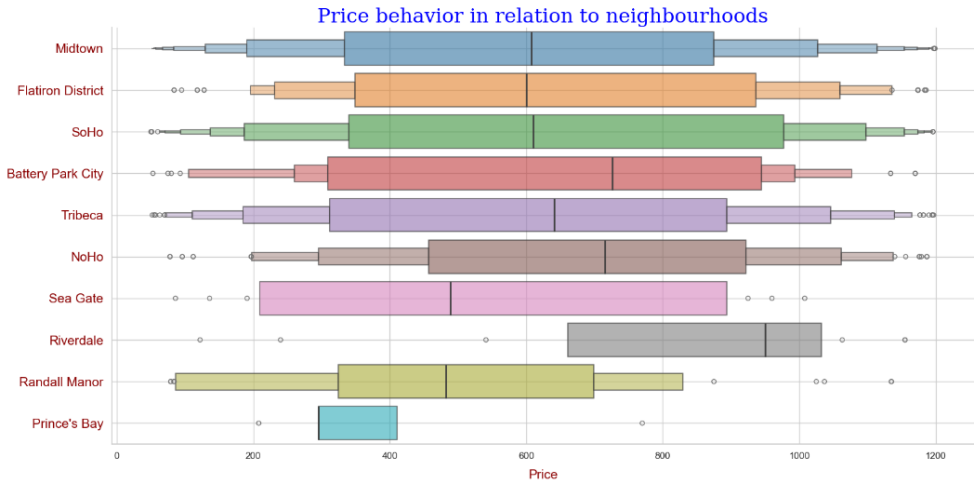


Figure 23. Multivariate Boxen Plot.

From the table and plot, Tribeca shows a broad range of prices with a high median, suggesting it's one of the more expensive neighborhoods. In contrast, Prince's Bay has a more modest price range and lower median price, indicating more affordable accommodation options. The variation within neighborhoods can be substantial, as seen in areas like Riverdale and Battery Park City, where the price spread from the lower to the upper quartile encompasses a wide range of values. This diversity in pricing within each neighborhood suggests that guests can find a variety of accommodation options, from budget-friendly to luxury, within the same area.

Table 6. Price Statistics by Neighbourhood

Price Statistics by neighbourhood					
neighbourhood	Min Price	25% Quantile	Median Price	75% Quantile	Max Price
Tribeca	$52.00	$311.50	$641.00	$894.00	$1197.00
Sea Gate	$86.00	$209.75	$489.00	$893.50	$1008.00
Riverdale	$122.00	$661.00	$950.50	$1033.00	$1155.00
Prince's Bay	$208.00	$296.00	$296.00	$411.00	$770.00
Battery Park City	$53.00	$309.00	$726.00	$944.25	$1170.00
Flatiron District	$84.00	$348.75	$600.00	$936.50	$1186.00
Randall Manor	$79.00	$325.00	$482.00	$699.00	$1135.00
NoHo	$78.00	$457.00	$715.00	$922.00	$1187.00
SoHo	$50.00	$340.00	$610.00	$977.00	$1196.00
Midtown	$51.00	$334.00	$607.00	$875.00	$1199.00

12. Area Plot:

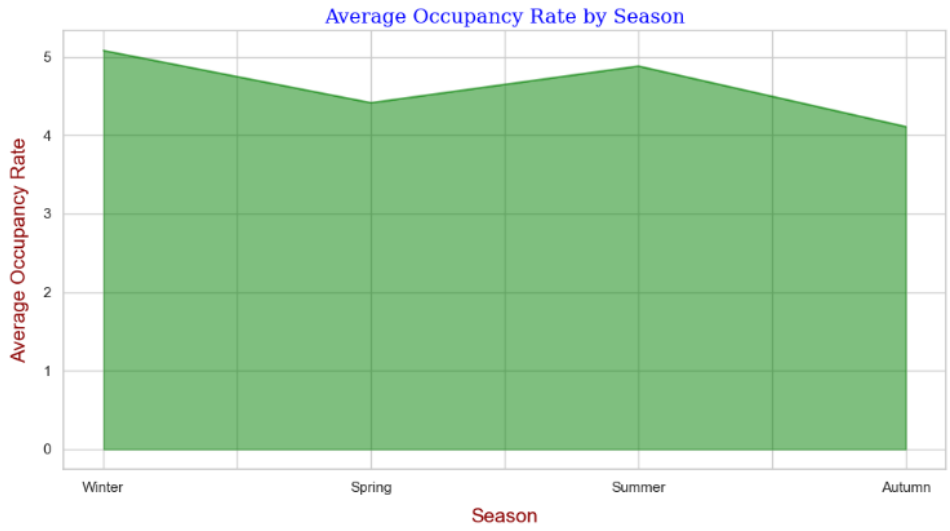


Figure 24. Area Plot.

The area chart illustrates the average occupancy rate for Airbnb listings across different seasons. The y-axis indicates the occupancy rate, while the x-axis represents the four seasons of the year. The area under the line is filled in, providing a clear visual representation of the relative occupancy rates throughout the year.

Table 7. Seasonal Occupancy Rates.

Seasonal Occupancy Rates	
Season	Average Occupancy Rate
Winter	5.08
Spring	4.41
Summer	4.88
Autumn	4.11

According to the chart and accompanying data table, the winter season shows the highest average occupancy rate at just over 5. This could be attributed to the holiday season and tourist attractions during the colder months. Spring sees a slight decline in the occupancy rate, with the average dropping to around 4.4, possibly due to a reduction in travel after the winter holidays. Summer picks up again slightly, with an average rate close to 4.9, which may reflect the peak tourist season and warmer weather encouraging more travel. Autumn shows the lowest occupancy rate, with the average falling to approximately 4.1, which may indicate a seasonal dip as temperatures drop and fewer holiday-driven travels occur. This pattern provides insights into how occupancy fluctuates with the seasons, which could help hosts in planning and pricing strategies throughout the year.

13. Violin Plot:

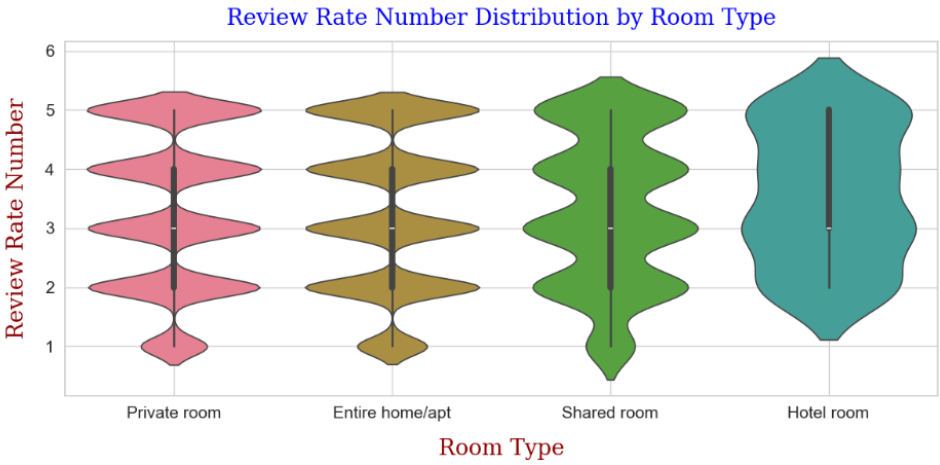


Figure 25. Violin Plot.

The violin plot illustrates the distribution of review rates across different room types on Airbnb. Each 'violin' shows the density of the data at different values, with wider sections corresponding to a higher density, meaning more data points fall in that range. The internal box plot within each violin highlights the median and interquartile ranges, providing a quick visual interpretation of the central tendency and spread.

Table 8. Review Rate Number Distribution by Room Type (Violin Plot)

Review Rate Number Distribution by Room Type								
Room Type	Count	Mean	Std	Min	25%	50%	75%	Max
Entire home/apt	44164	3.27	1.29	1.00	2.00	3.00	4.00	5.00
Hotel room	108	3.54	1.13	2.00	3.00	3.00	5.00	5.00
Private room	37494	3.29	1.28	1.00	2.00	3.00	4.00	5.00
Shared room	1646	3.30	1.24	1.00	2.00	3.00	4.00	5.00

According to the accompanying table, all room types have review rates ranging from 1 to 5, with medians around 3. This indicates a general tendency towards middle-range review ratings across all accommodations. Shared rooms have the highest average review rate at 3.30, followed closely by hotel rooms at 3.54, private rooms at 3.29, and entire homes/apartments at 3.27. The tight clustering of the medians and the similar shapes of the violins suggest there's a consistent pattern in guest satisfaction regardless of room type, with a slight edge for shared and hotel rooms in terms of average review rate. This visualization provides a comparative view of guest feedback trends, revealing subtle differences in the guest experience across various lodging options on the platform.

14. Joint Plot with KDE and Scatter Representation:

The joint plot integrates both a scatter plot and kernel density estimates (KDE) to offer a comprehensive view of the relationship between listing popularity and host response rate. The scatter plot, located at the center, demonstrates the distribution of individual listings with their corresponding popularity against the host's response rate.

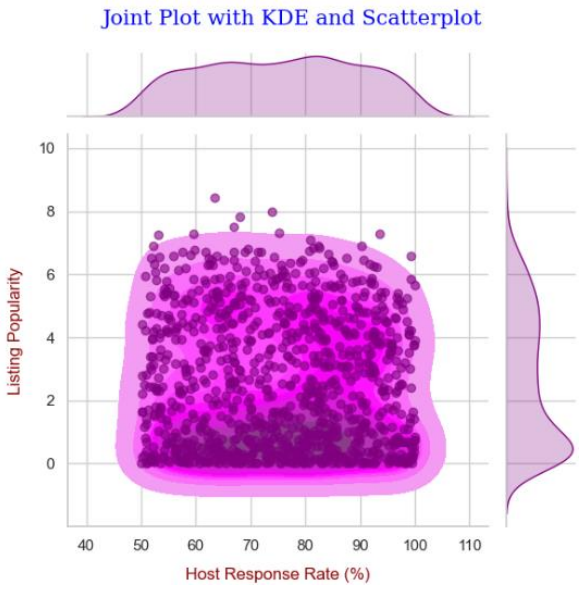


Figure 26. Joint Plot with KDE and Scatter Representation.

The marginal histograms on the top and right edges of the joint plot indicate the univariate distribution of each variable, with the horizontal histogram showing the distribution of host response rates and the vertical histogram presenting the distribution of listing popularity.

Table 9. Statistics of Host Response Rate and Listing Popularity.

Statistics of Host Response Rate and Listing Popularity		
Statistics	Host Response Rate	Listing Popularity
Mean	75.33	2.59
Median	75.88	2.26
Standard Deviation	14.13	2.11

As per the table provided, the average host response rate is approximately 75.33%, while the mean listing popularity score is around 2.59. The medians are closely aligned with the means, suggesting a symmetrical distribution around the central tendency for both variables. The standard deviation for host response rate is 14.13, which is significantly larger than the standard deviation for listing popularity at 2.11, indicating a wider variation in how promptly hosts respond compared to the variation in listing popularity scores.

15. Rug Plot:

A rug plot is a type of data visualization that is used to show the distribution of a single variable. It consists of a one-dimensional scatter plot, which is essentially a series of data points along an axis. Each point represents a single observation from a dataset. In a rug plot, these points are typically represented by lines or marks along the x or y-axis of a chart [7].

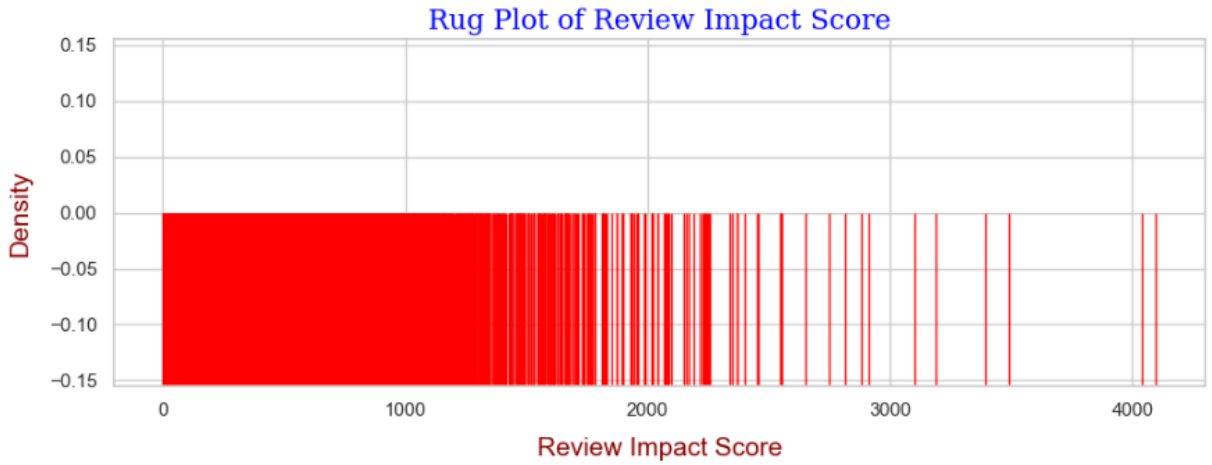


Figure 27. Rug Plot.

This rug plot offers a visual representation of the Review Impact Score across listings. The plot showcases individual data points as marks along the X-axis, providing an immediate sense of the data's distribution density. The accumulation of lines at the plot's base illustrates where scores are more concentrated, with sparser sections indicating less common score values. Notably, there is a clustering of data points near the origin, suggesting that many listings have a low Review Impact Score.

The Review Impact Score itself is a multiplicative metric, likely derived from the number of reviews and some measure of review quality or frequency, giving an indication of how much influence reviews might have on a listing's perceived quality or popularity. The plot indicates a right-skewed distribution, with a significant number of listings having scores on the lower end and a few extending to much higher values. This could imply that while most listings do not amass substantial review impact, there are outliers with exceptionally high scores, potentially due to either a large number of reviews, higher review rates, or a combination of both.

16. 3D Plot:

Figure 28. 3D Plot.

The 3D plot presents a multi-dimensional analysis of Airbnb prices in relation to both host activity scores and the number of reviews. Price, represented on the vertical axis, varies in accordance with the color gradient, creating a visual representation of price distribution across different levels of host activity and reviews. The plot indicates a complex relationship where higher host activity scores and an increasing number of reviews do not uniformly correspond to higher prices. Instead, there seems to be a wide dispersion of prices across the range of host activity scores, while the number of reviews appears to be more densely clustered.

Table 10. Summary Statistics for Key Metrics (3D Plot).

Summary Statistics for Key Metrics								
Metric	Count	Mean	Std	Min	25%	50% (Median)	75%	Max
Price	83412	$626.22	$331.79	$50.00	$340.00	$625.00	$914.00	$1200.00
Host Activity Score	83412	23.96	108.68	1.00	3.00	4.00	8.00	1660.00
Number of Reviews	83412	2.61	1.35	0.69	1.39	2.48	3.66	6.93

The accompanying table summarizes the data used for the 3D plot, providing a statistical overview of the three key metrics: price, host activity score, and number of reviews. With the mean price at approximately $626 and the mean host activity score around 24, there's a notable variation in the data, as indicated by the standard deviations. The table also suggests a broad price range from $50 to $1200, which reflects the diverse accommodation options available on the platform. The data also indicates that most hosts have a relatively low activity score and a modest number of reviews, with the median values for both metrics leaning towards the lower end of the scale.

17. Contour Plot:

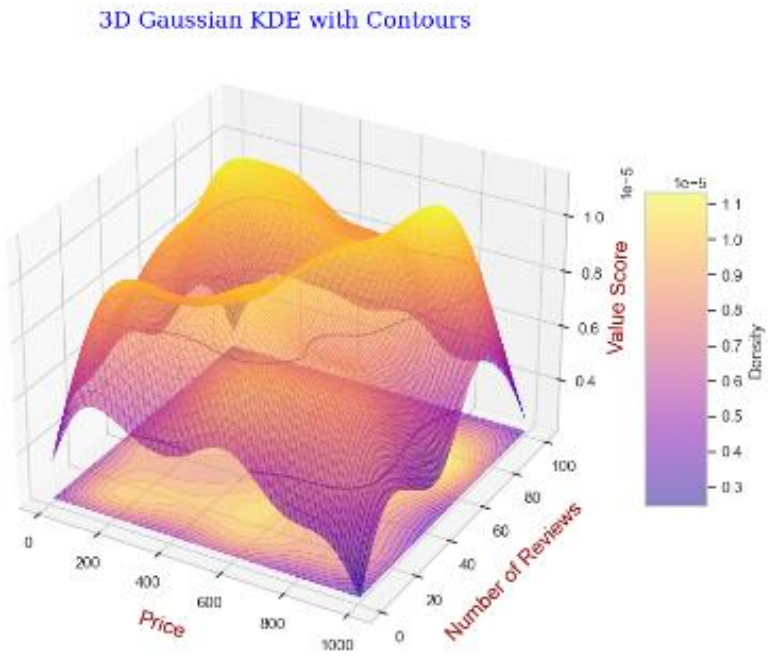


Figure 29. 3D Gaussian KDE with Contours.

The contour plot visualizes the density of data points in a 3D space, providing a Gaussian Kernel Density Estimate across price, the number of reviews, and an associated value score. Contours represent areas of equal density, with higher peaks indicating a greater concentration of data points and lower areas signifying sparser data. The color gradient, ranging from purple to yellow, denotes the progression from low to high-density regions. Yellow peaks suggest common combinations of price and number of reviews that are more prevalent in the dataset, while the purple troughs indicate less common combinations.

In this visualization, the concentration of points around the lower price range with a moderate number of reviews suggests that more affordable listings receive a consistent number of reviews. As the price increases, the density decreases, indicated by the contours spreading out, which could imply that higher-priced listings are less common or receive reviews less consistently. The plot is instrumental in understanding how price and reviews interact in relation to the value score, providing an overview of the market's behavior. By studying the contours, one can identify the price and review number thresholds that delineate the most densely populated regions, essential for understanding market dynamics.

18. Cluster Map:

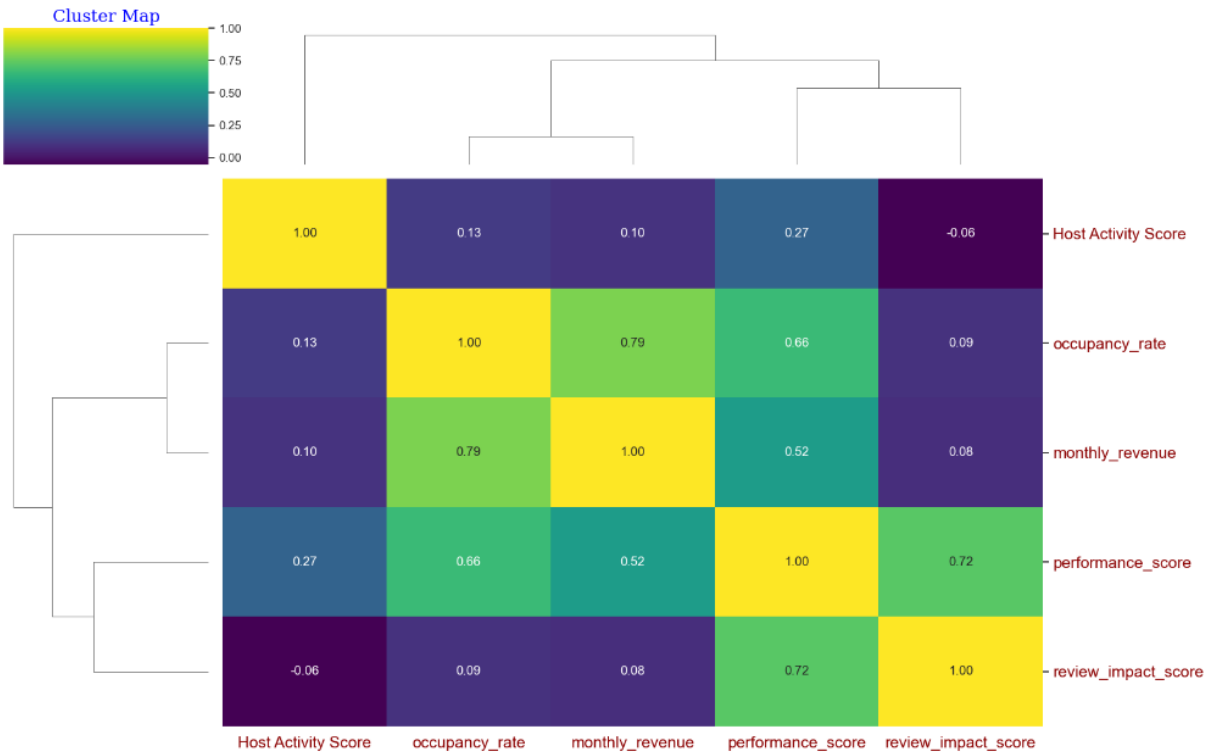


Figure 30. Cluster Map.

This cluster map visualizes the correlation between various metrics and clusters them based on similarity in correlation. It combines a heatmap with hierarchical clustering to group similar variables together. On the heatmap, the correlation coefficients range from -1 to 1, with warmer colors like yellow indicating a strong positive correlation and cooler colors like blue signifying a weaker or negative correlation.

From the heatmap, it's observed that monthly revenue and performance score have a relatively high positive correlation, as indicated by the yellow block with a value of 0.52, suggesting that as the monthly revenue increases, the performance score tends to rise as well. The occupancy rate also shows a strong positive relationship with monthly revenue, evidenced by a correlation coefficient of 0.79, signifying that listings with higher occupancy rates often generate more revenue. In contrast, the host activity score appears to have little to no linear relationship with the other metrics, highlighted by the cooler colors in its row and column. The dendrograms on the axes group variables with similar correlation patterns, providing a visual summary of how the metrics are interrelated.

19. Hexbin Plot:

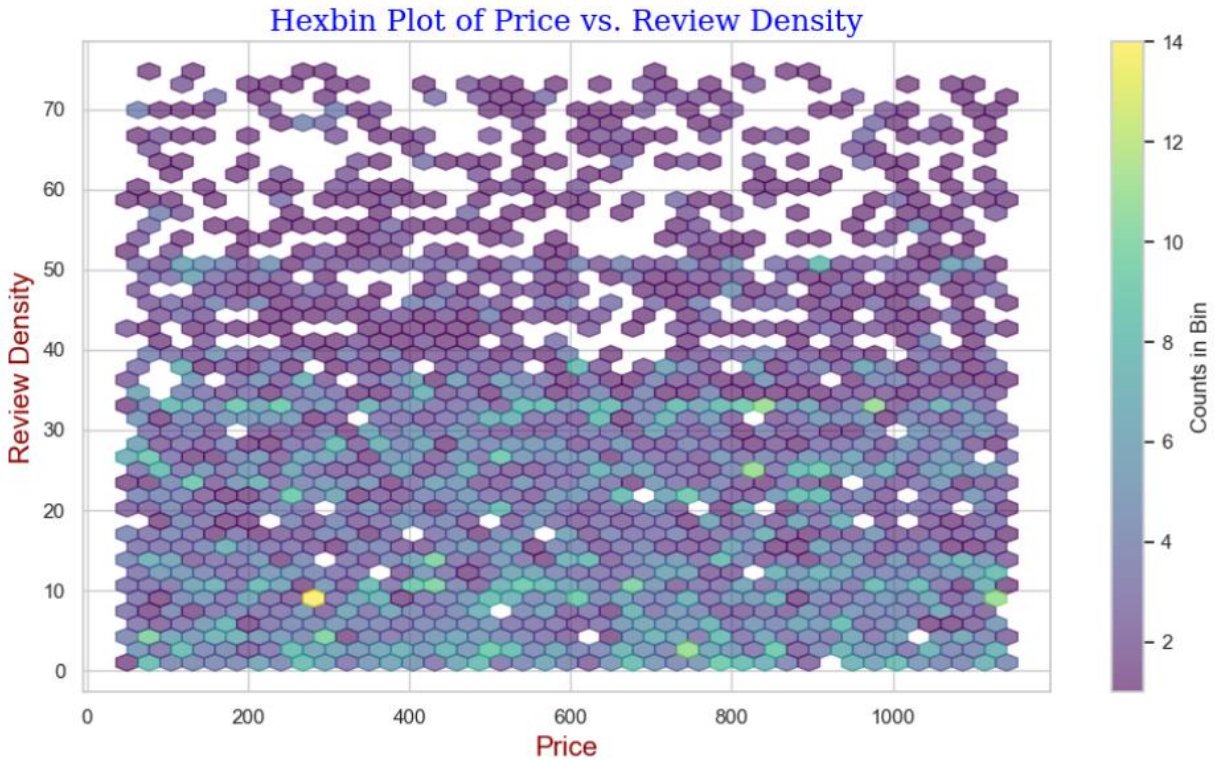


Figure 31. Hexbin Plot.

The hexbin plot presents a two-dimensional depiction of the relationship between price and review density using hexagonally-shaped bins. Each hexagon represents a bucket of points, with the color indicating the count of observations within that bin. The color bar to the right translates the color to the exact number of counts, revealing the density of points in each region of the plot.

In this particular hexbin plot, there's a notable concentration of data around the lower to mid-price range with moderate review densities, shown by the darker hexagons. The scarcity of data at higher price points and extremely high or low review densities is marked by lighter hexagons. This visual tool effectively conveys how price levels correlate with the frequency of reviews a listing receives, highlighting patterns and concentrations within the dataset. The few isolated hexagons with warmer colors suggest outliers or less frequent occurrences where listings at certain price points have unexpectedly high or low review densities.

20. Strip Plot:

The strip plot displays the distribution of prices for different types of rooms on Airbnb and differentiates them based on the verification status of the host's identity. Each point on the plot represents the price of a listing, with the type of room on the vertical axis and the price on the horizontal axis. Verified hosts are marked with a different color than unverified hosts, allowing for an easy comparison of pricing trends between the two groups across the different room types.

From the visualization, it's apparent that private rooms and entire homes/apartments are the most common room types offered, with private rooms generally priced lower than entire homes/apartments. Shared rooms show the lowest price points, indicating they are the most economical option on the platform.

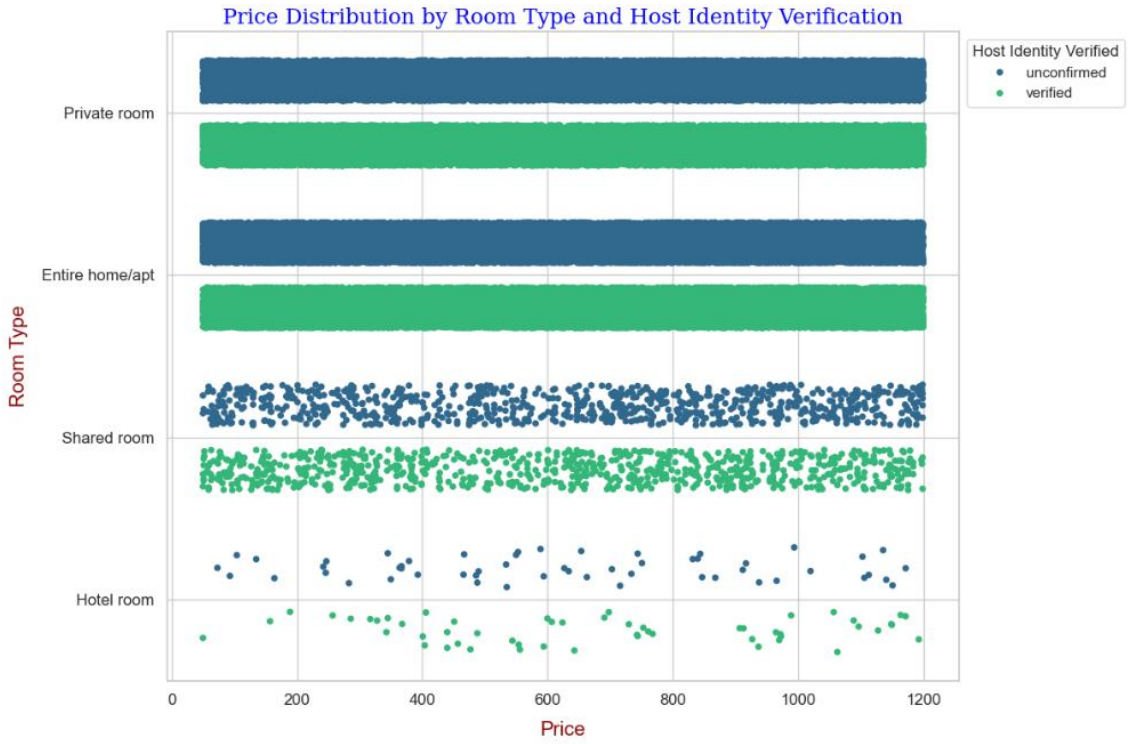


Figure 32. Strip Plot.

Hotel rooms appear to be less common but span a wide range of prices, potentially indicating a variety of offerings from budget to luxury. The verification status of the host's identity does not seem to significantly affect the price distribution within each room type category, as both verified and unverified hosts are spread throughout the price range.

21. Swarm Plot:

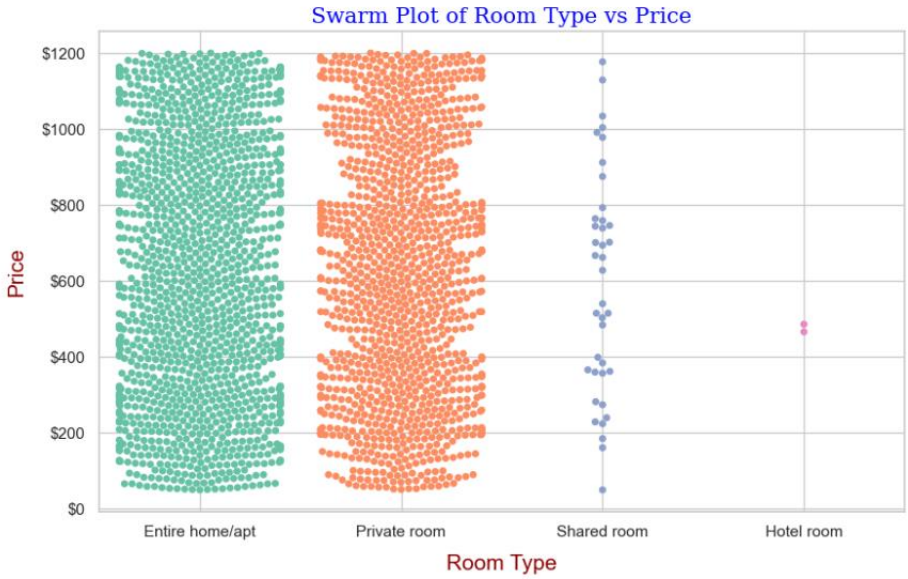


Figure 33. Swarm Plot.

The swarm plot illustrates the range of prices for various types of room listings on Airbnb, neatly organizing individual data points to show the distribution of prices within each category without overlap. This visualization method provides clarity on the density and distribution of pricing for entire homes/apartments, private rooms, shared rooms, and hotel rooms. Entire homes and private rooms display a broad range of prices stretching from the lower to the upper end, with a high concentration around the median price. Shared rooms and hotel rooms show a narrower spread in prices, and the hotel rooms, in particular, exhibit a tight cluster indicating less variation in pricing.

Table 11. Price Statistics by Room Type (Swarm Plot).

Price Statistics by Room Type			
Room Type	Minimum Price	Median Price	Maximum Price
Entire home/apt	$50	$610	$1200
Hotel room	$466	$476	$486
Private room	$51	$614	$1200
Shared room	$50	$628	$1177

The accompanying table provides a statistical summary of the price distribution for each room type, confirming the visual data shown in the swarm plot. The median prices offer a quick comparison of the central tendency in pricing across different room types, while the minimum and maximum prices give insights into the range of the dataset. For instance, the median prices for private rooms and shared rooms are quite close, suggesting a similar price point for both types of accommodations, despite shared rooms typically being a more budget-friendly option. The table also highlights that hotel rooms, while less varied, tend to have a higher entry price point compared to other room types.

Subplots

1. Airbnb Pricing Trends Over Two Decades (First Subplot).

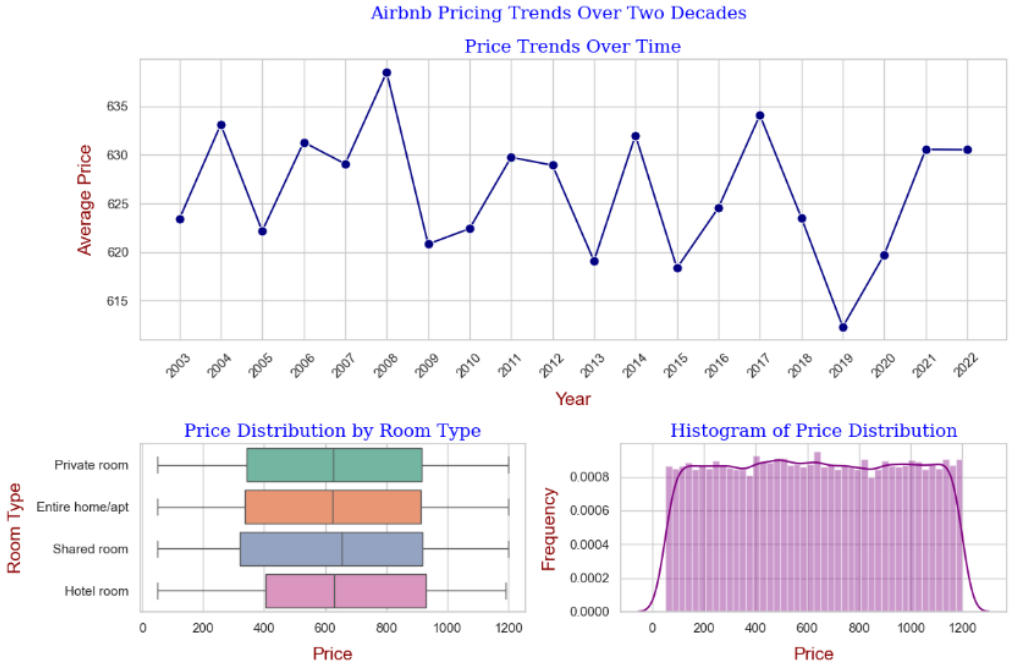


Figure 34. Airbnb Pricing Trends Over Two Decades.

The line graph titled "Price Trends Over Time" presents an overview of the average pricing changes for Airbnb listings across several years. The graph's oscillating nature with pronounced ups and downs suggests a volatile market where average prices fluctuate notably over time. Notable peaks in the graph indicate years when average pricing reached its zenith, possibly due to increased demand or economic growth, making those years potentially lucrative for hosts. On the other hand, the valleys following these peaks could reflect a decrease in average prices, which might be due to a variety of factors including off-peak seasons, increased competition, or economic downturns.

The visualization shows how prices in the sharing economy's accommodation sector have gone up and down over time. We can see big jumps in prices around 2007 and 2011, maybe because more people were visiting or there weren't enough places to stay. Then, after 2011, prices dropped, maybe because more places to stay were available or fewer people were traveling. The graph also helps us understand how things like events or changes in what people want affect prices over time.

In essence, this subplot not only charts historical data but also serves as a decision-making tool for both hosts deciding on pricing strategies and guests planning their travels. By identifying patterns and correlations in this historical data, stakeholders can make more informed predictions and adjustments for future pricing and booking strategies.

2. Trend Analysis (Second Subplot).

Figure 35. Trend Analysis.

The Trend Analysis subplot showcases a multifaceted view of Airbnb's market dynamics, focusing on two crucial aspects: host efficiency across different neighborhood groups and the evolution of review impact scores over time. The first bar graph, depicting "Average Host Efficiency by Neighbourhood Group," reveals a striking contrast among the regions. It illustrates that Manhattan hosts tend to be the most efficient, potentially indicating a higher guest turnover or better management practices. The other neighborhoods, such as Staten Island and the Bronx, show considerably lower average efficiency, which could be due to less tourist traffic or different hosting styles.

The line graph, "Trend of Review Impact Scores Over Time," indicates a general upward trajectory with some fluctuations in between the years observed. This trend suggests an increasing significance of reviews in the Airbnb ecosystem over the years, with the impact score soaring in recent years, highlighting a period where perhaps customer feedback has become more influential in the marketplace.

Overall, this subplot provides actionable insights into the hosting environment of different New York City neighborhoods and the escalating importance of reviews in the Airbnb community. For hosts and market analysts, such data is invaluable for strategizing on guest engagement and property management, particularly in Manhattan where the potential for high efficiency is evident. Meanwhile, the progressive increase in review impact scores underscores the necessity for hosts to maintain high standards and actively manage their online reputation to succeed in a review-centric market landscape.

3. Booking Features Distribution (Third Subplot).

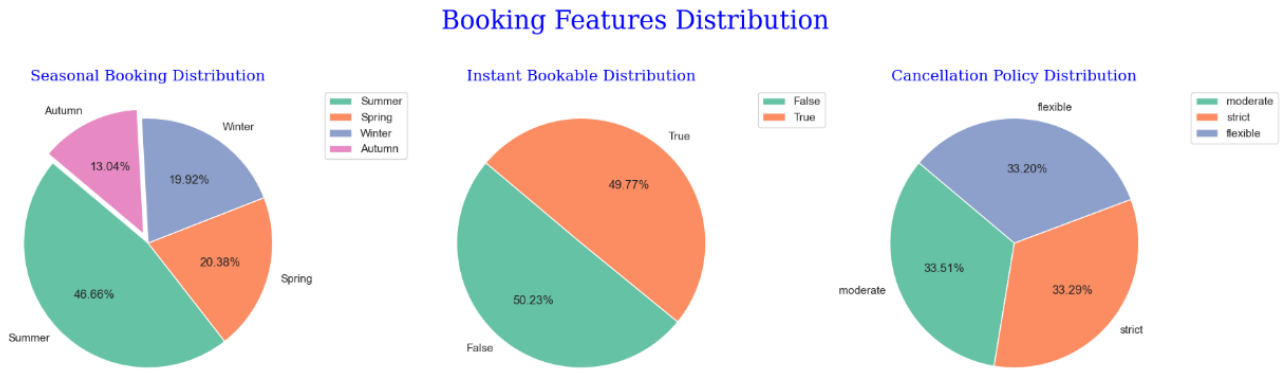


Figure 36. Booking Features Distribution.

The "Booking Features Distribution" subplot visually communicates the variances in booking preferences across three different dimensions within an accommodation-sharing platform. The first pie chart illustrates the 'Seasonal Booking Distribution,' where the largest segment belongs to summer bookings, comprising nearly half of all reservations, followed by spring. Winter and autumn account for the smallest share, indicating a potential off-peak period for hosts. This distribution highlights the seasonality effect in lodging preferences, with summer being the peak travel season likely due to favorable weather conditions and vacation times.

The middle pie chart, representing 'Instant Bookable Distribution,' splits almost evenly, showing a slight majority of listings are available for instant booking. This suggests a balanced preference among hosts, with just over half favoring the immediacy that caters to guests seeking quick booking confirmations, enhancing convenience and appeal to spontaneous travelers.

Lastly, the 'Cancellation Policy Distribution' pie chart shows a three-way split among flexible, moderate, and strict cancellation policies. Each category holds approximately one-third of the distribution, indicating no overwhelming preference for any particular policy. This balance signifies a diverse market with varied risk appetites, where hosts are equally distributed in their approach to cancellations, offering options to suit different guest preferences and planning horizons.

Collectively, these visual data points underscore the dynamic nature of the lodging market, highlighting the adaptability of hosts to meet diverse guest needs. For market strategists and hosts, understanding these preferences aids in optimizing listing features to align with traveler demands, especially focusing on the lucrative summer season and the convenience of instant booking.

4. Neighbourhood Group Dynamics (Fourth Subplot).

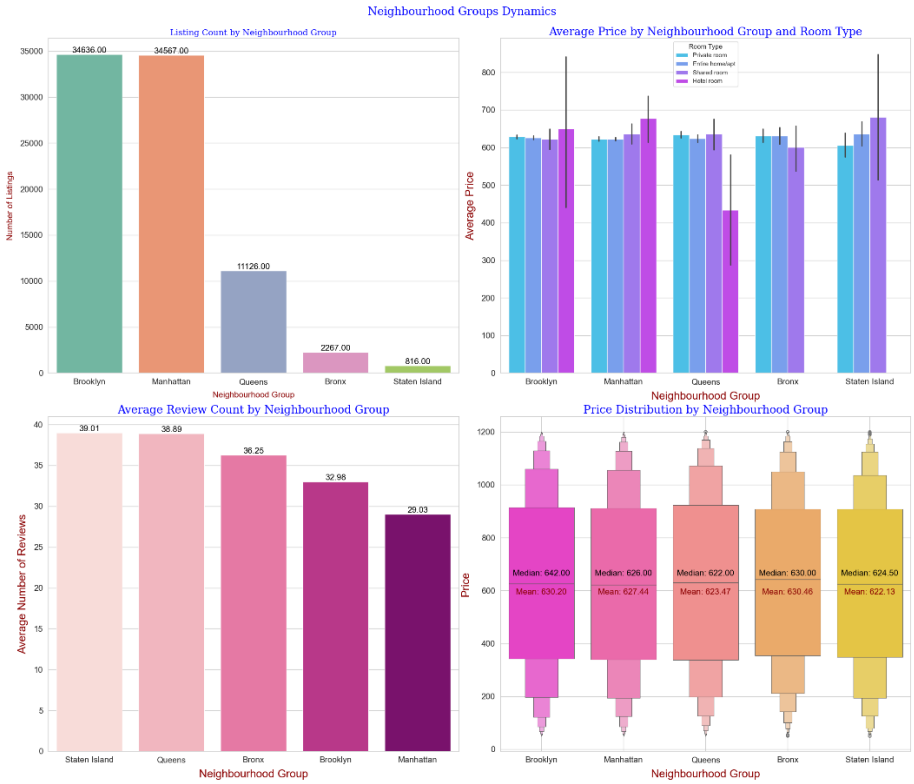


Figure 37. Neighbourhood Group Dynamics.

The fourth subplot, titled "Neighbourhood Group Dynamics" presents a comparative analysis of Airbnb prices across different neighbourhood groups. The data is displayed using boxen plots, which are an enhanced box plot designed to reveal more information about the distribution of data, particularly in large datasets.

The boxen plot provides a detailed view of the price ranges within each neighbourhood group, showing the median (central line inside the boxes), the interquartile ranges (edges of the boxes), and the overall distribution of prices through multiple quantiles (the extended tails of the plot). This visualization reveals the spread and skewness of prices within each group and identifies outliers that fall outside the typical range.

From the plot, it can be inferred that while there's a wide range of prices in each neighbourhood, certain groups tend to have higher median prices, as indicated by the central mark in each boxen. The length and symmetry of each plot provide insights into the variability and skewness of the price distribution. For example, the plot for Manhattan showcases a higher median price compared to other neighbourhoods, as well as a broader range indicating a diverse set of listings from low to high prices. Comparatively, the Bronx shows a more compact distribution, indicating less variability in Airbnb pricing within that area. Additionally, the presence of outliers is noticeable, particularly in neighbourhoods with extended tails, suggesting some listings are priced significantly higher than the typical range. This can reflect premium offerings in the market or potential anomalies in the data.

Dashboard

Google Cloud Platform Dashboard Link: https://dashapp-zxxyphmjaq-ue.a.run.app/

Dash is an open-source Python framework for building analytical web applications. It is developed and maintained by Plotly Technologies Inc [9]. Dash allows developers to create interactive, data-driven web applications using Python, without needing to write HTML, CSS, or JavaScript.

The Airbnb app interface comprises multiple tabs, each serving distinct functions and providing valuable insights into the Airbnb ecosystem.

The "Stays" tab acts as a gateway to accommodation-related information, offering access to listings, booking details, and property features. Users can seamlessly navigate through available stays, browse property photos, and make bookings directly from the app.

```
# ############################
# # PHASE 3 Layout
# ############################

app.layout = html.Div([
    html.Div([
        html.Img(src='assets/Airbnb_logo.jpeg', style={'width': '50px', 'height': '50px', 'vertical-align': 'middl
        html.H1("Airbnb",
                style={'display': 'inline-block', 'margin-left': '10px','margin-top': '20px', 'font-family': 'Time
    ], style={'text-align': 'center', 'margin-bottom': '20px'}),

    dcc.Tabs(id='tabs', value='tab-all',
             children=[
                 dcc.Tab(label='Stays', value='Stays', style={'border': 'none'}),
                 dcc.Tab(label='Normality Tests & Outlier Detection', value='NormOut', style={'border': 'none'}),
                 dcc.Tab(label='Statistical Analysis', value='Stats', style = {'border':'none'}),
                 dcc.Tab(label='Accomodation Trends', value='Trend', style={'border': 'none'}),
                 dcc.Tab(label='Sign up', value='Sign up', style = {'border': 'none'}),
             ],
             style={'border-bottom': '2px solid #FF464B', 'font-family': 'Times New Roman', 'font-size': '18px', '
             ),

    html.Div(id='tabs-content')
])
```

Figure 38. Code Snippet of App Layout.

In contrast, the "Normality Tests & Outlier Detection" tab houses powerful statistical tools designed to assess data distributions' normality and identify outliers. Through visualizations and statistical tests, users can thoroughly evaluate variables' distributional characteristics within the dataset, ensuring data quality and pinpointing potential anomalies with confidence.

Within the "Statistical Analysis" tab, users delve into comprehensive statistical summaries and analyses of the dataset. Here, they can explore descriptive statistics, conduct hypothesis testing, and unravel complex relationships between variables using advanced statistical techniques. This section empowers users to gain deeper insights into the dataset's characteristics and unveil underlying patterns and correlations.

Moving on to the "Accommodation Trends" tab, users uncover invaluable insights into the Airbnb accommodation market's evolving landscape. They can delve into historical data, visualize trends

over time, and dissect factors influencing accommodation demand and pricing dynamics. By leveraging interactive visualizations, trend analyses, and predictive modeling, users can forecast future trends and make informed decisions.

Lastly, the "Sign up" tab presents users with a convenient avenue to create accounts or sign in, granting access to personalized features. These include saving favorite listings, managing bookings, and receiving notifications about new listings and promotions. This section fosters user engagement and interaction with the Airbnb platform, ultimately enhancing the overall user experience. The following app was then deployed via Google Cloud Platform [8].

1. Stays Tab.

The first tab, "Stays," provides users with interactive tools to explore and filter Airbnb listings based on neighborhood groups, room types, and price ranges. Through dropdown menus, radio buttons, and a price range slider, users can dynamically adjust the displayed listings according to their preferences, enhancing the usability and customization of the application. The tab also includes descriptive titles and headings to provide context and clarity to the displayed data, ensuring a seamless and informative user experience.

In the first tab “Stays Layout”, the dropdown menu functionality is implemented using the dcc.Dropdown component, providing users with a straightforward selection of neighborhood groups such as Brooklyn, Manhattan, Queens, Bronx, and Staten Island. This enables users to filter the map display to show listings from specific areas, enhancing the user experience with convenient filtering options.

To select the type of room, dcc.RadioItems is utilized, offering clear choices including 'Private room', 'Entire home/apt', 'Shared room', and 'Hotel room'. This component dynamically updates the map based on the user's selection, ensuring that the displayed data accurately reflects the user's accommodation preferences.

The price range slider, implemented with dcc.RangeSlider, allows users to set a minimum and maximum price range interactively. This slider empowers users to refine their search results according to budget constraints, enhancing the usability of the application by providing a powerful filtering tool.

Descriptive titles and headings, added using HTML components like html.H1, provide clarity and context to the interface, improving the user experience by clearly communicating the purpose of the displayed data.

The map visualization is rendered using the dcc.Graph component, typically incorporating a Plotly map plot. This component enables interactive data visualization, displaying the spatial distribution of Airbnb listings color-coded by neighborhood group or price. It dynamically responds to user interactions with the dropdown, radio buttons, and slider, ensuring a seamless and responsive user experience.

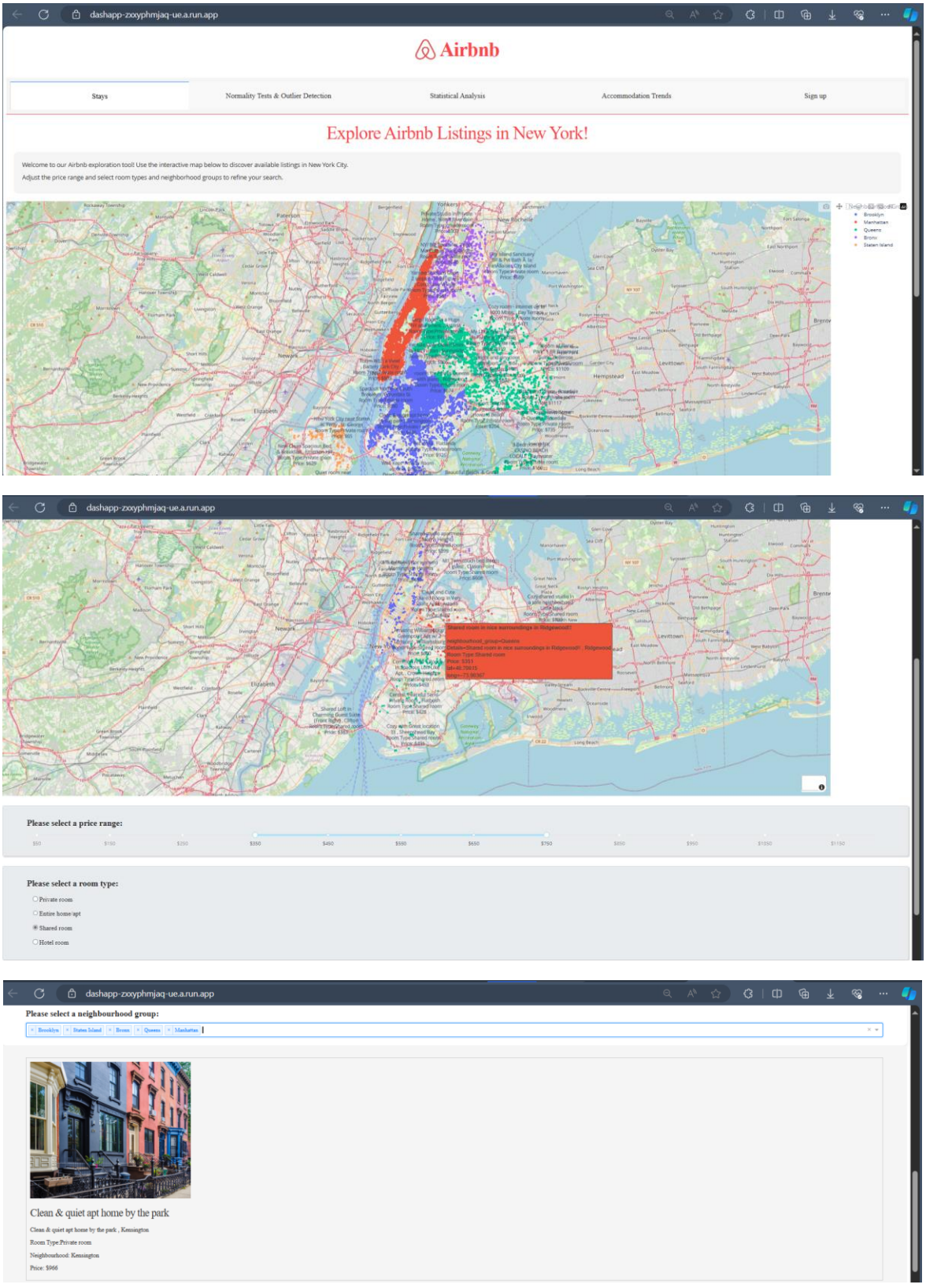


Figure 39. Stays (Tab 1).

The first tab presents a feature at the end that allows users to select multiple neighborhood groups simultaneously for more comprehensive search results. This functionality is provided by setting the multi=True property of the dcc.Dropdown component in Dash. With this property enabled, users aren't limited to selecting a single option; they can choose several neighborhoods to tailor their search more broadly.

When multiple neighborhood groups are selected, the app generates a list of properties from those specific areas. Each listing displayed includes detailed information such as the address, price, room type, and perhaps additional attributes like proximity to landmarks or a description of the home. This multi-select feature enhances the flexibility of the app, providing users with a richer set of choices and enabling them to compare listings across different areas.

By accommodating multiple selections, the app caters to users who may be considering several neighborhoods in their housing search or who want to explore different aspects of the market in New York City. The multi=True feature thus greatly expands the app's usability, offering customized and comprehensive insights into the variety of Airbnb options available.

Dash's layout design is inherently reactive and dynamic. When users interact with input components, Dash's callback functions update the output components, such as the map display, in real-time. This interactivity is a fundamental aspect of Dash's functionality, enabling the creation of sophisticated data-driven web applications accessible to Python programmers without extensive web development experience.

2. Normality Tests & Outlier Detection Tab.

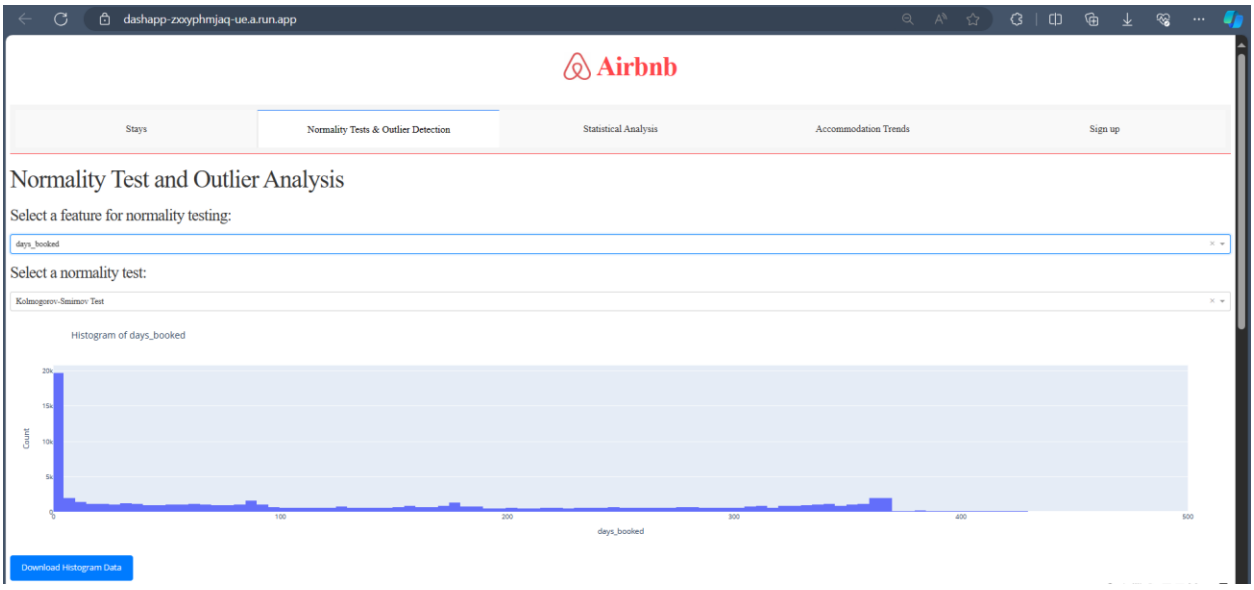


Figure 40. Normality Tests & Outlier Detection (Tab 2).

The "Normality Tests & Outlier Detection" tab of the Airbnb web application serves as a dedicated platform for conducting essential data preprocessing tasks, including normality testing and outlier analysis. These processes are fundamental in understanding the distribution of data and ensuring the accuracy of subsequent statistical analyses. Leveraging the Dash framework, this interface provides users with intuitive tools to assess data normality and detect outliers, empowering them to make informed decisions based on robust statistical foundations.

As observed in the outlier detection & removal and the normality test sections, this tab utilizes Dash's capabilities to create interactive visualizations and tools for data analysis. The layout is designed to facilitate user-friendly interactions, allowing users to explore and analyze the dataset effectively. By combining descriptive elements with interactive components, such as dropdown menus, sliders, and radio buttons, the interface enables users to conduct comprehensive data analysis tasks with ease and efficiency.

For the layout, Dash provides Python classes for all of the visual components of the application. In this specific tab, dcc.Loading is utilized, which is a Dash Core Components (dcc) module that provides a loading state to display to users when a component is being updated, thereby improving user experience during interactions that require a wait time for data processing or retrieval.

Another important Dash component in use here is dcc.Download which allows users to download data from the app. This can be particularly useful for downloading the results of statistical analyses, like the histogram data for days booked, providing users with the flexibility to use the data for further analysis or reporting outside of the Dash app.

Moreover, the histogram displaying the distribution of days booked is likely rendered using dcc.Graph, which is a component of Dash that creates interactive visualizations using Plotly. Users can interact with these graphs, zooming in on areas of interest or hovering over data points to get more information.

The application also includes interactive components like dropdowns and buttons, constructed with dcc.Dropdown and html.Button respectively, enabling users to select different variables and options for their analyses. The selection for normality testing can be changed using a dropdown, allowing the user to choose among different statistical tests such as the Kolmogorov-Smirnov Test.

3. Statistical Analysis Tab.

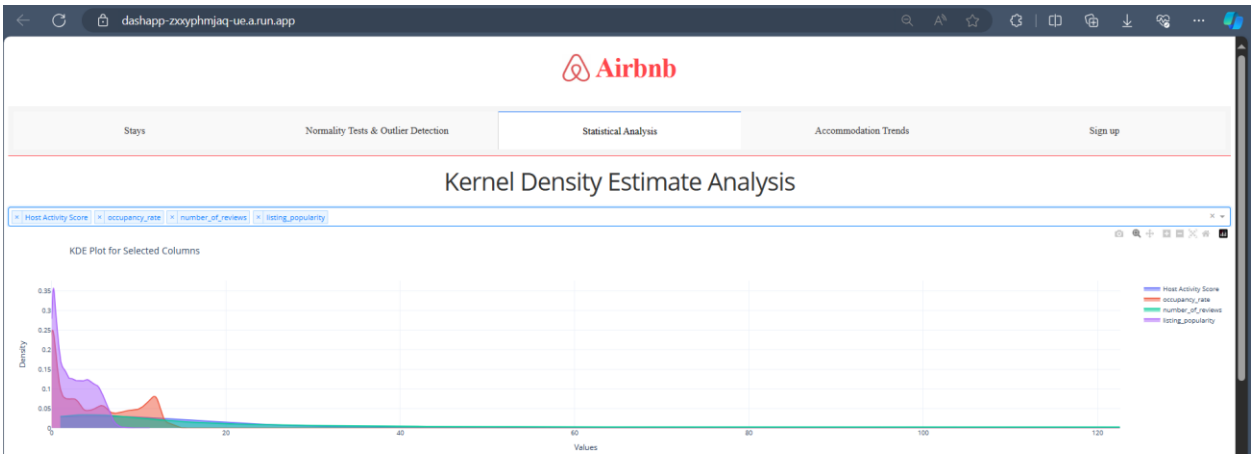


Figure 41. Statistical Analysis (Tab 3).

The "Statistical Analysis" tab in the Airbnb application serves as a powerful tool for visualizing the distribution and density of various dataset features. Constructed using Dash, an analytical web framework, this tab offers an interactive environment where users can explore different features for analysis via a dropdown menu created with dcc.Dropdown.

The implementation employs dcc.Graph to generate a multivariate Kernel Density Estimate (KDE) graph, facilitating the visualization of the probability density of multiple variables concurrently. This feature provides valuable insights into data structure and relationships that may not be immediately apparent when examining individual variables in isolation.

To enhance user experience, the dcc.Loading component wraps the graph, effectively managing the loading state of the application during graph rendering or updates. This feature signals to users that data processing is underway, managing expectations while reducing perceived waiting times.

The integration of Dash and its components, such as dcc.Dropdown and dcc.Graph, enables extensive customization and interaction. Users can dynamically select features for analysis, and the KDE plot adjusts accordingly, displaying density estimations for selected features like "Host Activity Score," "occupancy_rate," "number_of_reviews," and "listing_popularity."

This tab's inclusion within a broader statistical analysis section underscores its role in providing a comprehensive analytical toolkit. Leveraging Dash's flexibility and seamless integration with Python's data science libraries like Pandas and Plotly, the application facilitates statistical testing, data visualization, and interactive data exploration, empowering users to derive meaningful insights from the dataset.

4. Accommodation Trends Tab.

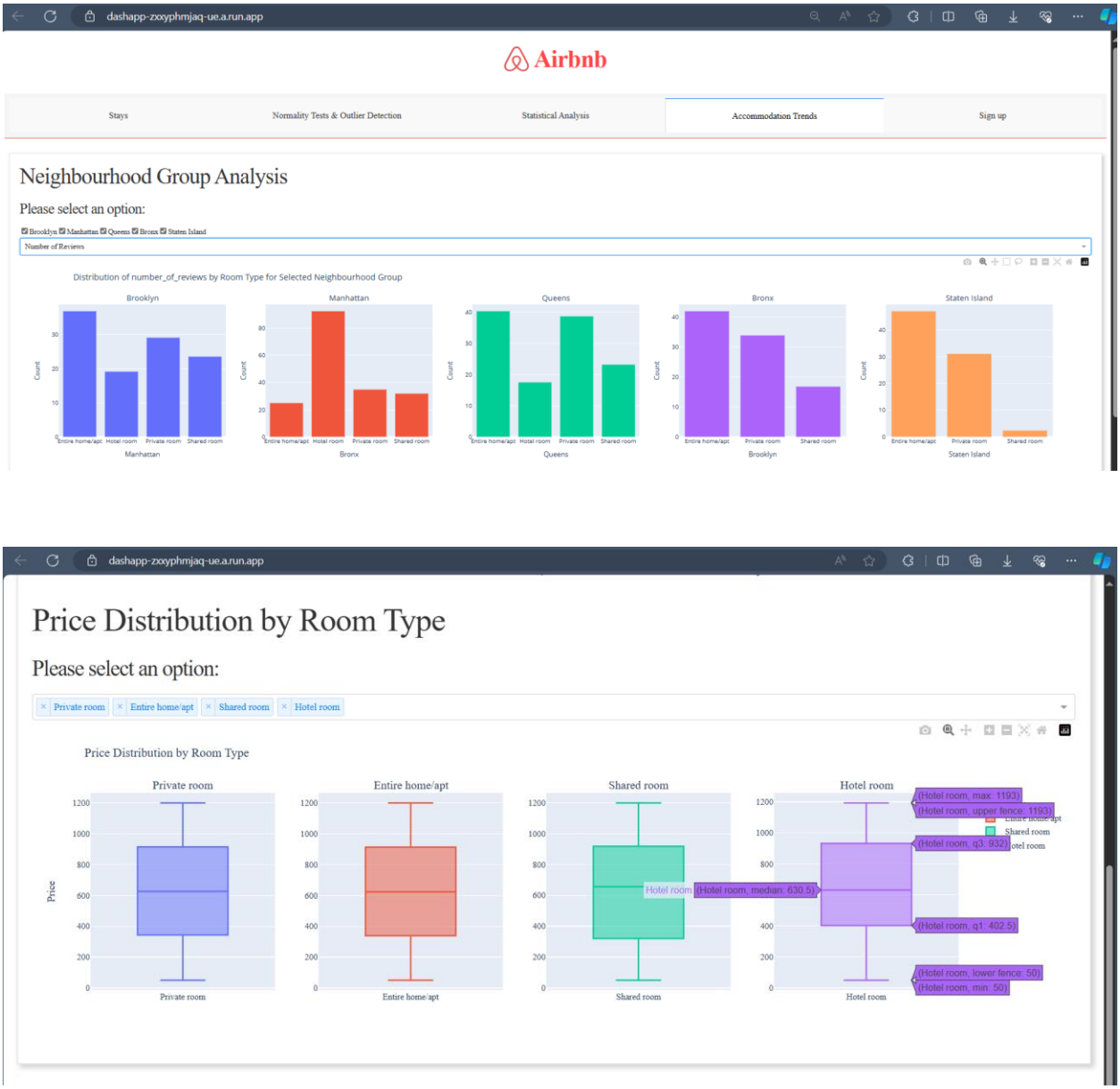


Figure 42. Accommodation Trends (Tab 4).

The "Accommodation Trends" tab on the Airbnb dashboard serves as an analytical tool to explore the distribution of prices across different types of accommodations. It leverages several components from Dash, a Python framework for building interactive web applications, and Plotly, a graphing library that enables the creation of sophisticated plots.

In this tab, the dcc.Checklist component is employed to allow users to select multiple options from a list, which, in this case, represents different types of room accommodations such as 'Private

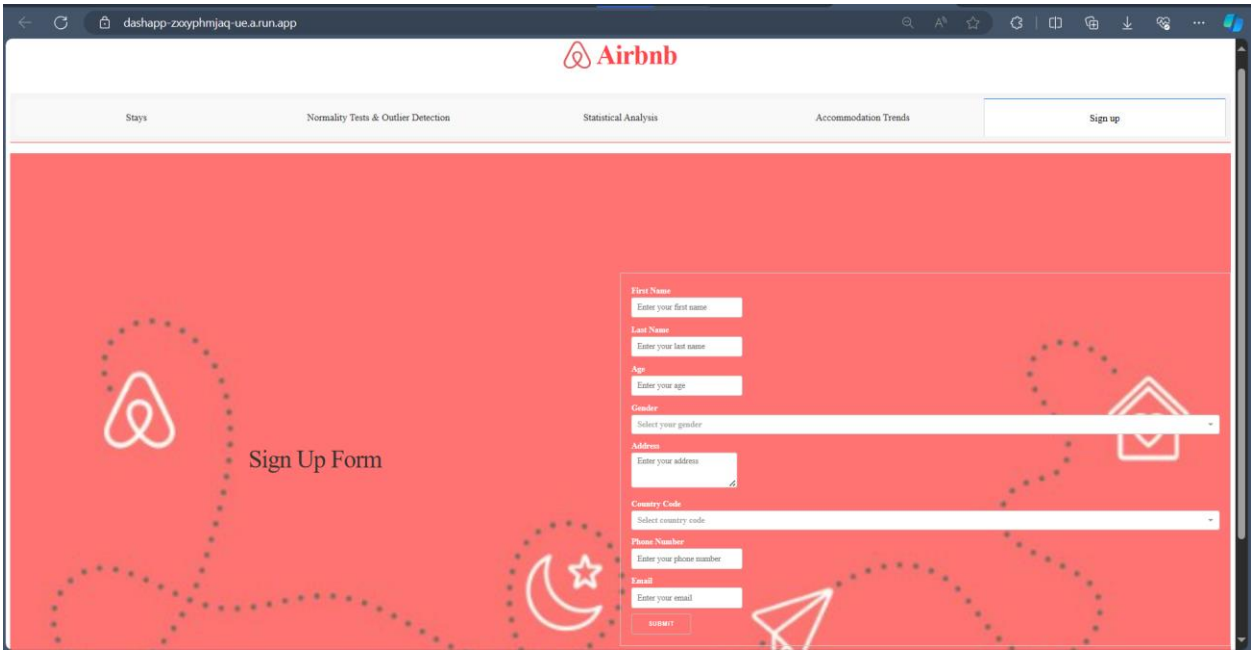
room', 'Entire home/apt', 'Shared room', and 'Hotel room'. When a user selects one or more room types, the selection triggers an update to the visualizations displayed on the page.

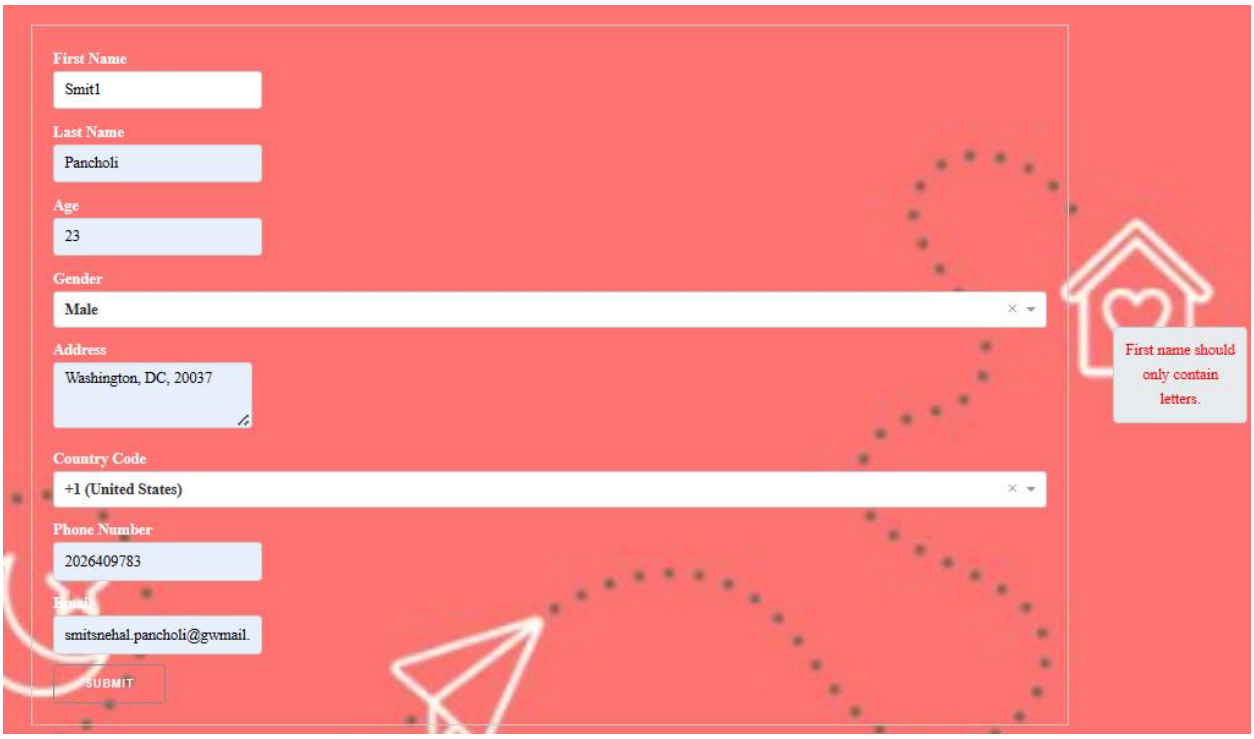
The html.Br component is a simple line break, used here to ensure proper visual spacing between different elements of the UI, which helps to keep the application clean and easy to navigate.

The dcc.Graph component is utilized to display interactive data visualizations. It is highly versatile and capable of rendering complex plots. In this particular tab, the make_subplots function from Plotly is used to create a composite figure that combines multiple subplots into a single visual display. This allows for a more compact and comparative presentation of data. In this context, the function likely generates a series of box plots to illustrate the price distribution for each room type, enabling users to compare median prices, variability, and the presence of outliers across different accommodation options.

The "Accommodation Trends" tab is designed to provide an insightful and interactive way for users to investigate pricing patterns in the Airbnb market, empowering them with the information needed to make informed decisions, whether they are booking a stay or listing a property. The use of Dash and Plotly ensures that the data is not only accurately represented but also presented in a way that is engaging and user-friendly.

5. Sign Up Tab.


dashapp-zoxyphmjaq-ue.a.run.app
Airbnb
Stays
Normality Tests & Outlier Detection
Statistical Analysis
Accommodation Trends
Sign up
Sign Up Form
First Name
Enter your first name
Last Name
Enter your last name
Age
Enter your age
Gender
Select your gender
Address
Enter your address
Country Code
Select country code
Phone Number
Enter your phone number
Email
Enter your email
SUBMIT


First Name
Smit1
Last Name
Pancholi
Age
23
Gender
Male
Address
Washington, DC, 20037
Country Code
+1 (United States)
Phone Number
2026409783
smitsnehal.pancholi@gwmail.
SUBMIT
First name should only contain letters.

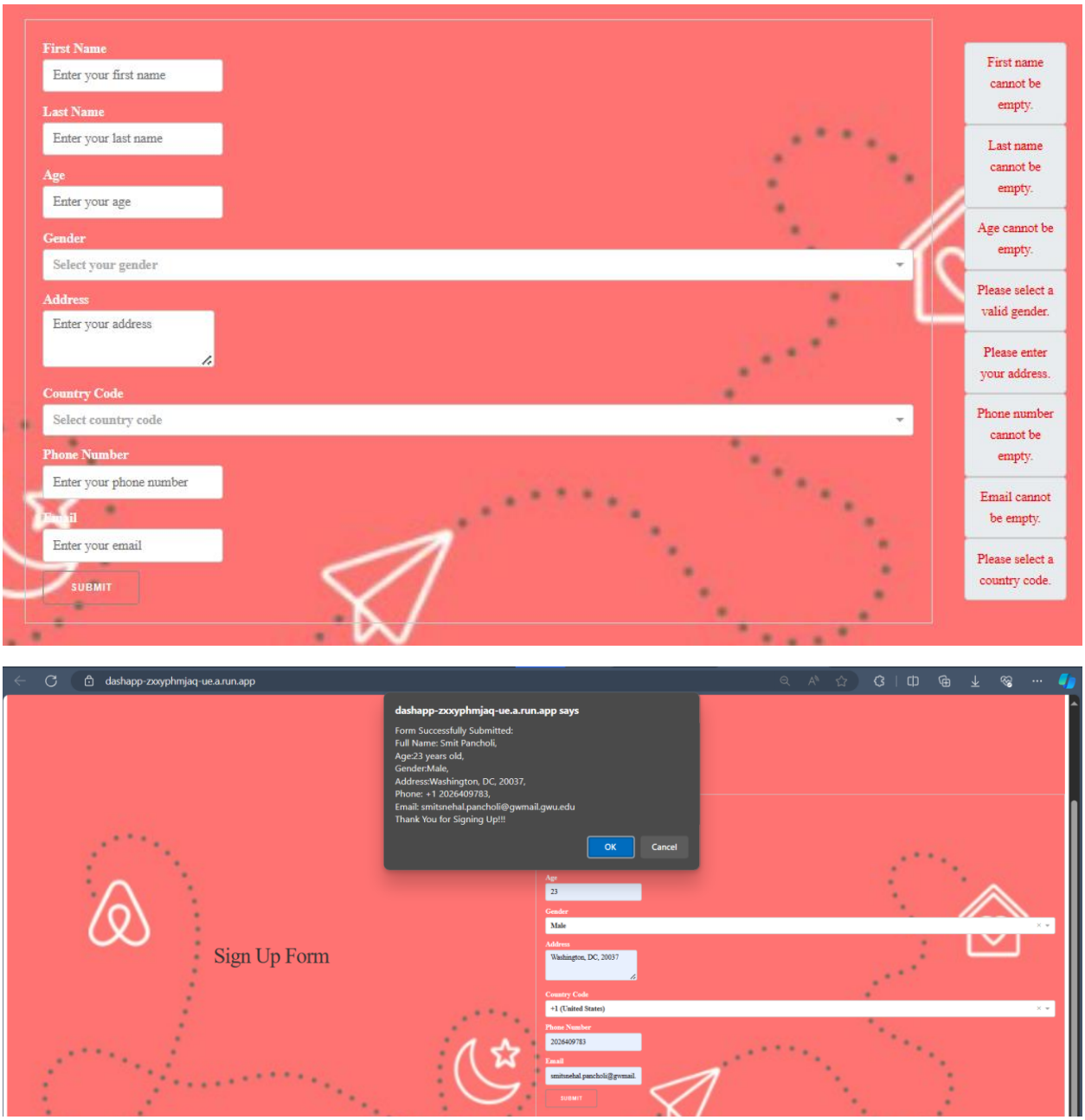


Figure 43. Sign Up (Tab 5).

The sign-up form in the tab “Sign Up”, is a pivotal component of the user interface meticulously crafted using Dash, a robust Python framework tailored for constructing web analytic applications. Dash harnesses the power of React.js, offering a plethora of rich and interactive user components. Within this form, specific Dash components such as dcc.Input, html.Button, and dcc.Dropdown play pivotal roles in gathering user data. Each component fulfills a distinct function, from capturing text input to facilitating selection from a dropdown menu.

Input fields are seamlessly integrated using dcc.Input, imperative for soliciting crucial information like name, age, and contact details. Accompanying these input fields are html.Label elements, furnishing clear descriptions for each input to guide users on the required information. The country code selection is streamlined through a dcc.Dropdown, presenting a predefined list of country codes for standardized and error-free entries. To streamline the gender field, dcc.RadioItems might be employed, furnishing users with predefined options to ensure data consistency.

Augmenting the input mechanisms, validation messages are tactfully displayed using dcc.Tooltip. This component provides immediate feedback to users, presenting contextual messages adjacent to fields that require attention. Such instant feedback not only reinforces data quality but also enhances the user experience by mitigating frustration associated with incorrect submissions.

Enclosed within an html.Div container, these components are meticulously organized, providing structure to the web page while facilitating additional styling and positioning of form elements. Placeholders embedded within dcc.Input components offer users cues about the expected data format, further enhancing the user interface by preempting invalid inputs.

The submission action is flawlessly executed using the `html.Button` component, triggering the validation process upon user click. Upon successful validation, a `dcc.ConfirmDialog` or a JavaScript alert window is employed to convey a confirmation message, reaffirming the successful submission and expressing gratitude for the user's sign-up.

Conclusion

The analysis reveals clear trends dictating the Airbnb scene in New York City (NYC). The high concentration of listings in Manhattan and Brooklyn indicates a strong preference among hosts for these vibrant boroughs, likely due to their central location and easy accessibility. The popularity of entire homes/apartments, with a median price of $625, highlights guests' desire for privacy and spacious accommodations, catering to both tourists and business travelers.

Neighborhoods like Williamsburg and Bedford-Stuyvesant emerge as popular destinations, boasting numerous reviews across all rating scales. This signifies their appeal to guests and reflects a thriving hospitality scene in these areas. Despite their lower numbers, hotel rooms garner the highest average review rates, indicating elevated guest satisfaction within this accommodation category.

The wide range in Airbnb pricing challenges the notion that it is always more expensive than traditional lodging, with options available for as low as $50. This pricing diversity caters to various guest budgets and needs.

While host activity scores show a weak correlation with monthly revenue, occupancy rates strongly influence host earnings. Verified listings, particularly hotel rooms, command slightly higher prices, suggesting guests may value the added security and authenticity.

a. Insights from Graphical Analysis:

The graphical analysis through various plots in this project has been instrumental in unveiling multifaceted perspectives within the Airbnb dataset. PCA illustrated the data's dimensionality, emphasizing the variance within fewer components and thereby streamlining complex datasets for advanced analyses. Normality tests provided critical insights into the data distribution, indicating where standard statistical models might fall short and alternative approaches would be more suitable. Correlation matrices, scatter plots, and other visual tools vividly detailed the relationships among variables, informing potential areas of focus, such as the influence of occupancy rates over performance scores.

b. Utility of the Dashboard:

The dashboard serves as an analytical hub, facilitating a deep dive into the dataset with a user-centric approach. Interactive elements like sliders and dropdowns offer real-time data manipulation, allowing users to customize their analysis and derive insights immediately. The dashboard brings data to life, transforming numbers into visual stories that are easier to understand and act upon. By providing comprehensive statistical computations, it also negates the need for external tools, streamlining the analytical workflow.

c. User-Friendly Assessment:

User experience design is at the heart of the app, ensuring accessibility for diverse users through intuitive design and structured navigation. The 'Stays' tab, with its dynamic map and multi-selection capabilities, exemplifies the app's commitment to a seamless user experience. It reflects an understanding that the utility of data is not just in its analysis but also in its presentation and accessibility. The sign-up form, with its structured input fields and real-time validations, furthers the app's user-friendly nature, guiding users with clarity and ease.

d. Functionality:

Functionality is the cornerstone of the app, manifested through a variety of tabs each catering to different analytical needs. From statistical analyses to trend exploration and personal account management, the app offers a comprehensive suite of tools that support a wide array of functions essential for engaging with the Airbnb marketplace. It is a testament to the app's robustness and its ability to provide a one-stop solution for users seeking to analyze, understand, and leverage data in the context of Airbnb's dynamic market.

In conclusion, the project's graphical analyses provide a vital foundation for understanding complex data structures and relationships within the Airbnb market. The dashboard is a testament to the power of interactive data visualization and user-centric design, offering an intuitive platform for data exploration and analysis. The app stands out for its functional depth, user-friendly interface, and holistic approach to addressing the varied analytical requirements of its users. These qualities not only make the app an indispensable tool for users but also demonstrate the potential of thoughtful design in unlocking the power of data.

Appendix

Python Code of Phase I (Static Plots).

```
#%% Importing Libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from prettytable import PrettyTable
import statsmodels.api as sm
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from matplotlib.gridspec import GridSpec
from scipy.stats import gaussian_kde


#%%
pd.set_option('display.max_columns', None)

#%% Importing the dataset and setting the style

df = pd.read_csv('Airbnb_open_data.csv', encoding='latin1')
print(df.head(5))

sns.set(style="whitegrid")

#%% Checking for Null Values

df.isnull().sum()
#%%

columns_to_drop = ['license', 'house_rules']
df_clean = df.drop(columns=columns_to_drop, errors='ignore')
df_clean.info()

#%%
df_clean.dropna(inplace = True)
print("df type:", type(df_clean))
print("df shape:", df_clean.shape)

#%%

before_duplicates = len(df_clean)
df_clean.drop_duplicates(inplace=True)
after_duplicates = len(df_clean)

print(f"Total number of records before removing duplicates:
{before_duplicates}")
print(f"Total number of records after removing duplicates:
{after_duplicates}")

#%%
```

```
df_clean.rename(columns={'availability 365': 'days_booked'}, inplace=True)
df_clean.columns = df_clean.columns.str.lower().str.replace(' ', '_')

df_clean['price'] = df_clean['price'].replace({'\$': '', ',': ''},
regex=True)
df_clean['service_fee'] = df_clean['service_fee'].replace({'\$': '', ',':
''}, regex=True)

df_clean['price'] = pd.to_numeric(df_clean['price'], errors='coerce')
df_clean['service_fee'] = pd.to_numeric(df_clean['service_fee'],
errors='coerce')

#%%
df_clean['neighbourhood_group'].unique()


#%%
corrections = {
    'brookln': 'Brooklyn',
    'manhatan': 'Manhattan'
}
df_clean['neighbourhood_group'] =
df_clean['neighbourhood_group'].replace(corrections)
print(df_clean['neighbourhood_group'].unique())

#%% Cleaned Dataframe

print(df_clean.head(5))

#%%

room_type_counts = df_clean ['room_type'].value_counts()
print("Count of Various Room Types Available:")
print(room_type_counts)

#%%

strict_cancellation =
df_clean.groupby('room_type')['cancellation_policy'].value_counts().groupby(l
evel=0).head(1)
print("Room type with the most strict cancellation policy:")
print(strict_cancellation)

#%%
avg_price_neighbourhood =
df_clean.groupby('neighbourhood_group')['price'].mean().sort_values(ascending
=False)
print("Average price per neighbourhood group:")
print(avg_price_neighbourhood)

#%% Principal Component Analysis (PCA)

numerical_cols = df_clean.select_dtypes(include=['int64',
'float64']).columns.tolist()

scaler = StandardScaler()
```

```
df_scaled = scaler.fit_transform(df_clean[numerical_cols])

pca = PCA()
principalComponents = pca.fit(df_scaled)


singular_values = [f"{sv:.2f}" for sv in pca.singular_values_]
cond_number = np.linalg.cond(df_scaled)

print('Singular Values of Original Feature:\n', singular_values)
print('\nCondition Number of Original Feature:\n', f"{cond_number:.2f}")

cumulative_variance_ratio = np.cumsum(pca.explained_variance_ratio_)

print("\nExplained variance ratio of original feature space:")
print([f"{var:.2f}" for var in pca.explained_variance_ratio_])

explained_variance_ratio = [f"{var:.2f}" for var in
pca.explained_variance_ratio_]
explained_variance_cols = [numerical_cols[i] for i in
range(len(explained_variance_ratio))]

print("\nFeatures before PCA\n",explained_variance_cols)
num_features_to_keep = np.argmax(cumulative_variance_ratio >= 0.95) + 1
num_features_to_remove = df_scaled.shape[1] - num_features_to_keep

print("\nNumber of features to remove:", num_features_to_remove)

pca = PCA(n_components=num_features_to_keep)
pca.fit(df_scaled)

cumulative_variance_ratio = np.cumsum(pca.explained_variance_ratio_)
num_components = range(1, len(cumulative_variance_ratio) + 1)
explained_variance_ratio = [f"{var:.2f}" for var in
pca.explained_variance_ratio_]
total_variance_explained = f"{sum(pca.explained_variance_ratio_):.2f}"

reduced_df_scaled = pca.transform(df_scaled)

singular_values_reduced = [f"{sv:.2f}" for sv in pca.singular_values_]
cond_number_reduced = np.linalg.cond(reduced_df_scaled)
print('Singular Values of Reduced Feature:\n', singular_values_reduced)
print('\nCondition Number of Reduced Feature:\n',
f"{cond_number_reduced:.2f}")

print("\nExplained variance ratio of reduced feature space:")
print([f"{var:.2f}" for var in pca.explained_variance_ratio_])

explained_variance_cols = [numerical_cols[i] for i in
range(len(explained_variance_ratio))]

print("\nFeatures after PCA\n",explained_variance_cols)

plt.figure(figsize=(14, 10))
plt.plot(num_components, cumulative_variance_ratio * 100, linestyle='-',
color='brown')
```

```
plt.axhline(y=95, color='red', linestyle='--', label='95% Explained
Variance')

optimal_idx = np.argmax(cumulative_variance_ratio >= 0.95) + 1

plt.axvline(x=optimal_idx, color='black', linestyle='--', label='Optimal
Number of Components')
plt.xlabel('Number of Components',fontname='serif', color='darkred', size=20)
plt.ylabel('Cumulative Explained Variance (%)',fontname='serif',
color='darkred', size=20)
plt.title('Cumulative Explained Variance vs. Number of
Components',fontname='serif', color='blue', size=20)
plt.yticks(np.arange(10, 103, 10))
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
table = PrettyTable(title="PCA Analysis Summary")

table.field_names = ["Metric", "Value"]

table.add_row(["Explained Variance Ratio", ',
'.join(explained_variance_ratio)])
table.add_row(["Total Variance Explained", total_variance_explained])
table.add_row(["Singular Values", ', '.join(singular_values_reduced)])
table.add_row(["Condition Number", f"{cond_number_reduced:.2f}"])
table.align["Metric"] = "l"
table.align["Value"] = "l"

print(table)


#%% Feature Engineering

average_price_per_room_type = df_clean.groupby('room_type')['price'].mean()
avg_price_neighbourhood = df_clean.groupby('neighbourhood')['price'].mean()

sorted_neighbourhoods_expensive =
avg_price_neighbourhood.sort_values(ascending=False)

top_10_expensive_neighbourhoods = sorted_neighbourhoods_expensive.head(10)

top_10_cheapest_neighbourhoods = sorted_neighbourhoods_expensive.tail(10)

df_clean_expensive_neighbourhoods =
top_10_expensive_neighbourhoods.reset_index()
df_clean_cheapest_neighbourhoods =
top_10_cheapest_neighbourhoods.reset_index()

print("Top 10 expensive neighbourhoods:")
for index, row in df_clean_expensive_neighbourhoods.iterrows():
    print(f"{row['neighbourhood']}: ${row['price']:.2f}")

print("\nTop 10 cheapest neighbourhoods:")
for index, row in df_clean_cheapest_neighbourhoods.iterrows():
    print(f"{row['neighbourhood']}: ${row['price']:.2f}")
```

```
df_clean['last_review'] = pd.to_datetime(df_clean['last_review'],
errors='coerce')

# 1. Monthly Revenue
df_clean['monthly_revenue'] = df_clean['price'] * df_clean['days_booked']

# 2. Occupancy Rate
df_clean['occupancy_rate'] = df_clean['days_booked'] / 30

# 3. Average Daily Rate (ADR)
df_clean['average_daily_rate'] = df_clean['monthly_revenue'] /
df_clean['days_booked']

# 4. Review Impact Score
df_clean['review_impact_score'] = df_clean['number_of_reviews'] *
df_clean['review_rate_number']

# 5. Seasonal Bookings (simplified to quarters)
df_clean['seasonal_booking'] = pd.cut(df_clean['last_review'].dt.month,
                                      bins=[0, 3, 6, 9, 12],
                                      labels=['Winter', 'Spring', 'Summer',
'Autumn'],
                                      right=False)

# 6. Host Efficiency
df_clean['host_efficiency'] = df_clean['calculated_host_listings_count'] /
df_clean['number_of_reviews']
df_clean['host_efficiency'] = df_clean['host_efficiency'].replace(np.inf, 0)

# 7. Host Activity Score
df_clean['Host Activity Score'] = df_clean['calculated_host_listings_count']
* df_clean['review_rate_number']

df_clean['reviews_frequency'] = np.log1p(df_clean['reviews_per_month'])

df_clean['norm_days_booked'] = (df_clean['days_booked'] -
df_clean['days_booked'].min()) / (df_clean['days_booked'].max() -
df_clean['days_booked'].min())
df_clean['norm_review_impact_score'] = (df_clean['review_impact_score'] -
df_clean['review_impact_score'].min()) /
(df_clean['review_impact_score'].max() -
df_clean['review_impact_score'].min())
df_clean['norm_host_efficiency'] = (df_clean['host_efficiency'] -
df_clean['host_efficiency'].min()) / (df_clean['host_efficiency'].max() -
df_clean['host_efficiency'].min())

df_clean['performance_score'] = 0.4 * df_clean['norm_days_booked'] + 0.4 *
df_clean['norm_review_impact_score'] + 0.2 * df_clean['norm_host_efficiency']


df_clean['host_efficiency_neighbourhood'] =
df_clean.groupby('neighbourhood_group')['host_efficiency'].transform('mean')

# Calculate the trend of review impact scores over time
df_clean['review_impact_trend'] =
df_clean.groupby(df_clean['last_review'].dt.year)['review_impact_score'].tran
sform('mean')
```

```
df_clean['year'] = df_clean['last_review'].dt.year


#
df_clean['reviews_per_month'] = df_clean['reviews_per_month'].replace(0,
np.nan)
df_clean['review_density'] = df_clean['number_of_reviews'] /
df_clean['reviews_per_month']
df_clean['reviews_count'] = np.log1p(df_clean['number_of_reviews'])
df_clean['reviews_frequency'] = np.log1p(df_clean['reviews_per_month'])

df_clean['host_response_rate'] = np.random.uniform(50, 100, len(df_clean))
df_clean['listing_popularity'] = np.log1p(df_clean['number_of_reviews'] *
df_clean['reviews_per_month'])

#%%

statistics = df_clean.describe()

# Print the statistics
print(statistics)
#%% HEATMAP WITH CBAR

numeric_cols = df_clean[['listing_popularity', 'reviews_count',
'occupancy_rate', 'performance_score']]
correlation_matrix = numeric_cols.corr()

plt.figure(figsize=(12, 8))
cmap = sns.diverging_palette(250, 10, as_cmap=True)

heatmap = sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap=cmap,
cbar=True,
                      annot_kws={"size": 16, "color": "black"},
                      cbar_kws={"shrink": 0.75})

cbar = heatmap.collections[0].colorbar
cbar.ax.tick_params(labelsize=10)

plt.title('Correlation Heatmap', fontsize=18, weight='bold',
color='blue',fontname='serif')
plt.xticks(rotation=0, fontsize=12, weight='bold')
plt.yticks(fontsize=12, weight='bold')

plt.tight_layout()
plt.grid(True)
plt.show()

sns.set_palette("Set2")
numeric_cols = df_clean[['listing_popularity', 'reviews_count',
'occupancy_rate', 'performance_score']]
sns.pairplot(numeric_cols)



plt.show()

table = PrettyTable(title="Correlation Matrix")
```

```
table.field_names = [""] + correlation_matrix.columns.tolist()

for index, row in correlation_matrix.iterrows():
    formatted_row = [f"{value:.2f}" for value in row.values]
    table.add_row([index] + formatted_row)

for column in table.field_names:
    table.align[column] = "r"

print(table)

################## DATA VISUALIZATION ###############
#%% LINE PLOT

construction_year_counts =
df_clean['construction_year'].value_counts().sort_index()

plt.figure(figsize=(10, 6))
plt.plot(construction_year_counts.index, construction_year_counts.values,
marker='o', markersize=8, linestyle='-', color='navy')
plt.title('Room Construction Trends (2003-2022)', fontname='serif',
color='blue', size=18, pad=20)
plt.xlabel('Year', fontname='serif', color='darkred', size=16, labelpad=20)
plt.ylabel('Rooms Built', fontname='serif', color='darkred', size=16,
labelpad=20)
plt.xticks(range(int(min(construction_year_counts.index)),
int(max(construction_year_counts.index)) + 1), size=13, rotation=45)
plt.yticks(size=13)
plt.grid(True, linestyle='--', linewidth=0.5, color='gray')
plt.tight_layout()
plt.show()


#%% COUNT PLOT

top_neighbourhoods = df_clean["neighbourhood_group"].value_counts()

plt.figure(figsize=(12, 8))
sns.countplot(x="neighbourhood_group",
hue="neighbourhood_group",data=df_clean, palette = 'Set2')

for i, count in enumerate(top_neighbourhoods.values):
    plt.text(i, count + 10, str(count), ha='center', va='bottom',
fontsize=15, fontweight='bold')

plt.title('NYC Airbnb Neighbourhood Groups ', fontname='serif', color='blue',
size=20, pad=15)
plt.xlabel('Neighbourhood group', fontname='serif', color='darkred', size=20,
labelpad=15)
plt.ylabel('Number of Airbnbs', fontname='serif', color='darkred', size=20,
labelpad=15)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.tight_layout()
plt.grid(True)
plt.show()
```

```
#%% BAR PLOT - GROUP PLOT

sorted_df = df_clean.groupby(['neighbourhood_group',
'room_type'])['review_rate_number'].mean().reset_index()
order =
sorted_df.groupby('neighbourhood_group')['review_rate_number'].mean().sort_va
lues(ascending=False).index

plt.figure(figsize=(15, 6))
sns.barplot(data=sorted_df, x='neighbourhood_group', y='review_rate_number',
hue='room_type', errorbar=None, order = order, palette='PuBu')
plt.title('Average Review Rate by Room Type and Neighbourhood Group',
fontname='serif', color='blue', size=20, pad=20)
plt.xlabel('Neighbourhood Group', fontname='serif', color='darkred', size=18,
labelpad=20)
plt.ylabel('Average Review Rate', fontname='serif', color='darkred', size=18,
labelpad=20)
plt.legend(title='Room Type', bbox_to_anchor=(1.02, 1), loc='upper left',
fontsize='large', title_fontsize='15')
plt.xticks(rotation=0, fontsize=15)
plt.yticks(fontsize=15)
plt.tight_layout()
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()

#%% BAR PLOT - STACKED PLOT
neighbourhood_counts = df['neighbourhood'].value_counts()

top_neighbourhood_counts = neighbourhood_counts.head(7)

filtered_df =
df_clean[df_clean['neighbourhood'].isin(top_neighbourhood_counts.index)]

stacked_data = filtered_df.groupby(['neighbourhood',
'review_rate_number']).size().unstack()

palette = sns.color_palette("Set3")

ax = stacked_data.plot(kind='bar', stacked=True, figsize=(10, 6),
color=palette)

plt.title('Review Rate Distribution by Neighbourhood (Top 7)',
fontname='serif', color='blue', size=15)
plt.xlabel('Neighbourhood', color='darkred', size=15, labelpad=15)
plt.ylabel('Count', color='darkred', size=15, labelpad=15)
plt.xticks(rotation=0)
plt.legend(title='Review Rate Number')
plt.tight_layout()
plt.grid(True)
for n, bar in enumerate(ax.patches):
    h = bar.get_height()
    if h != 0:
        ax.text(
            bar.get_x() + bar.get_width() / 2,
            bar.get_y() + h / 2,
            f'{int(h)}',
            ha='center',
```

```
            va='center',
            color='black',
            fontsize=8
        )

plt.show()

table = PrettyTable(title="Review Rate Distribution by Neighbourhood (Top
7)")
column_names = ["Neighbourhood"] + [f"Rate {rate}" for rate in
stacked_data.columns]
table.field_names = column_names

for neighbourhood, row in stacked_data.iterrows():
    row_data = [neighbourhood] + row.fillna(0).tolist()
    table.add_row(row_data)

table.align = "r"
table.align["Neighbourhood"] = "l"
print(table)
#%% PIE CHART

explode = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0.1)

neighbourhood_counts = df['neighbourhood'].value_counts()

top_neighbourhood_counts = neighbourhood_counts.head(10)

plt.figure(figsize=(10, 6))
plt.rcParams['axes.prop_cycle'] = plt.cycler(color=plt.cm.tab10.colors)

# Plot the pie chart
plt.pie(top_neighbourhood_counts, labels=top_neighbourhood_counts.index,
autopct='%1.2f%%', startangle=140, explode=explode, labeldistance=1.03)
plt.title('Top 10 Neighbourhood Distribution', fontname='serif',
color='blue', size=15)
plt.axis('equal')
plt.tight_layout()
plt.grid(True)
plt.show()


#%% DIST PLOT

plt.figure(figsize=(10, 6))
sns.distplot(df_clean['price'], kde=True, color='blue')

plt.title('Distribution of Airbnb Prices', fontname='serif', color='blue',
size=20)
plt.xlabel('Price', fontname='serif', color='darkred', size=20, labelpad=15)
plt.ylabel('Density', fontname='serif', color='darkred', size=20,
labelpad=15)

plt.tight_layout()
plt.grid(True)
plt.show()
```

```
price_stats = df_clean['price'].describe()

table = PrettyTable(title="Descriptive Statistics of Airbnb Prices")

table.field_names = ["Statistic", "Value"]

table.add_row(["Count", f"{price_stats['count']:.0f}"])
table.add_row(["Mean", f"${price_stats['mean']:.2f}"])
table.add_row(["Std", f"${price_stats['std']:.2f}"])
table.add_row(["Min", f"${price_stats['min']:.2f}"])
table.add_row(["25%", f"${price_stats['25%']:.2f}"])
table.add_row(["50% (Median)", f"${price_stats['50%']:.2f}"])
table.add_row(["75%", f"${price_stats['75%']:.2f}"])
table.add_row(["Max", f"${price_stats['max']:.2f}"])

table.align = "l"
table.align["Value"] = "r"

print(table)

#%% Histogram plot with KDE

sns.set(style="whitegrid")
filtered_data = df_clean[(df_clean['days_booked'] <= 500) &
(df_clean['days_booked'] > 0)]
stats = filtered_data['days_booked'].describe()


plt.figure(figsize=(12, 8))
sns.histplot(filtered_data['days_booked'], kde=True, color="red", bins=100)

plt.xlim(-10, 500)

plt.yscale('log')

plt.title('Distribution of Days Booked (Up to 500) with
KDE',fontname='serif', color='blue', size=15)
plt.xlabel('Days Booked',color='darkred', size=15, labelpad=10)
plt.ylabel('Frequency', color='darkred', size=15, labelpad=10)

ticks = plt.gca().get_yticks()

plt.gca().set_yticklabels([f"{int(y)}" if y >= 1 else f"{y:.2f}" for y in
ticks])

plt.xticks(range(0, 501, 50))

plt.tight_layout()
plt.grid(True)
plt.show()

table = PrettyTable(title="Descriptive Statistics for Days Booked (Up to 500
Days)")

table.field_names = ["Statistic", "Value"]

table.add_row(["Count", f"{stats['count']:.0f}"])
```

```
table.add_row(["Mean", f"{stats['mean']:.2f}"])
table.add_row(["Std", f"{stats['std']:.2f}"])
table.add_row(["Min", f"{stats['min']:.0f}"])
table.add_row(["25%", f"{stats['25%']:.0f}"])
table.add_row(["50% (Median)", f"{stats['50%']:.0f}"])
table.add_row(["75%", f"{stats['75%']:.0f}"])
table.add_row(["Max", f"{stats['max']:.0f}"])

table.align["Statistic"] = "l"
table.align["Value"] = "r"

print(table)

#%% QQ-plot

fig = sm.qqplot(df_clean['price'], line='s')
plt.title('QQ Plot of Price', fontname='serif', color='blue', size=15)
plt.xlabel('Theoretical Quantiles', color='darkred', size=15, labelpad=10)
plt.ylabel('Sample Quantiles', color='darkred', size=15, labelpad=10)
plt.tight_layout()
plt.grid(True)
plt.show()


#%% Airbnb Pricing Trends Over Two Decades (First subplot):

fig = plt.figure(figsize=(12, 8))

gs = GridSpec(2, 2, figure=fig, width_ratios=[1, 1], height_ratios=[1, 0.6])

ax1 = fig.add_subplot(gs[0, :])
ax2 = fig.add_subplot(gs[1, 0])
ax3 = fig.add_subplot(gs[1, 1])

years = np.arange(2003, 2023)
df_clean['construction_year'] = pd.to_numeric(df_clean['construction_year'],
errors='coerce')
df_yearly_price =
df_clean.groupby('construction_year')['price'].mean().reset_index()
sns.lineplot(x='construction_year', y='price', data=df_yearly_price, ax=ax1,
color='navy',marker='o', markersize=8, linestyle='-')
ax1.set_title('Price Trends Over Time',fontname='serif', color='blue',
size=15)
ax1.set_xticks(years)
ax1.set_xticklabels(years, rotation=45)
ax1.set_xlabel('Year',color='darkred', size=15, labelpad=10)
ax1.set_ylabel('Average Price',color='darkred', size=15, labelpad=10)
ax1.grid(True)

sns.boxplot(x='price', y='room_type', data=df_clean, hue =
'room_type',ax=ax2, palette='Set2')
ax2.set_title('Price Distribution by Room Type',fontname='serif',
color='blue', size=15)
ax2.set_xlabel('Price',color='darkred', size=15, labelpad=10)
ax2.set_ylabel('Room Type',color='darkred', size=15, labelpad=10)

sns.distplot(df_clean['price'], kde=True, ax=ax3, color='purple')
```

```
ax3.set_title('Histogram of Price Distribution',fontname='serif',
color='blue', size=15)
ax3.set_xlabel('Price',color='darkred', size=15, labelpad=10)
ax3.set_ylabel('Frequency',color='darkred', size=15, labelpad=10)

plt.suptitle('Airbnb Pricing Trends Over Two Decades', fontname='serif',
color='blue', size=15, x=0.55, y=0.98)
plt.tight_layout()
plt.show()

#%% Kernel Density Estimate with fill

df_clean['norm_days_booked'] = (df_clean['days_booked'] -
df_clean['days_booked'].min()) / (df_clean['days_booked'].max() -
df_clean['days_booked'].min())
df_clean['norm_review_impact_score'] = (df_clean['review_impact_score'] -
df_clean['review_impact_score'].min()) /
(df_clean['review_impact_score'].max() -
df_clean['review_impact_score'].min())
df_clean['norm_host_efficiency'] = (df_clean['host_efficiency'] -
df_clean['host_efficiency'].min()) / (df_clean['host_efficiency'].max() -
df_clean['host_efficiency'].min())

df_clean['performance_score'] = 0.4 * df_clean['norm_days_booked'] + 0.4 *
df_clean['norm_review_impact_score'] + 0.2 * df_clean['norm_host_efficiency']

plt.figure(figsize=(12, 6))
sns.kdeplot(df_clean['performance_score'], fill=True, alpha=0.6, linewidth=4,
palette='viridis')
plt.title('Kernel Density Estimate of Performance Score',fontname='serif',
color='blue', size=15)
plt.xlabel('Performance Score',color='darkred', size=15, labelpad=10)
plt.ylabel('Density',color='darkred', size=15, labelpad=10)
plt.grid(True)
plt.show()


#%% Trend Analysis (Second Subplot)


fig, axs = plt.subplots(2, 2, figsize=(20, 15))

host_efficiency =
df_clean.groupby('neighbourhood_group')['host_efficiency'].mean().sort_values
()
sns.barplot(x=host_efficiency.index, y=host_efficiency.values, ax=axs[0, 0],
palette = 'Set2')
axs[0, 0].set_title('Average Host Efficiency by Neighbourhood Group',
fontname='serif', color='blue', size=15)
axs[0, 0].set_xlabel('Neighbourhood Group', color='darkred', size=15)
axs[0, 0].set_ylabel('Average Host Efficiency', color='darkred', size=15)
axs[0, 0].tick_params(axis='x', rotation=0)

df_clean['year'] = df_clean['last_review'].dt.year
review_impact_trend = df_clean.groupby('year')['review_impact_score'].mean()
axs[0, 1].plot(review_impact_trend.index, review_impact_trend.values,
color='blue', marker='o', linestyle='-')
```

```
axs[0, 1].set_title('Trend of Review Impact Scores Over Time',
fontname='serif', color='blue', size=15)
axs[0, 1].set_xlabel('Year', color='darkred', size=15)
axs[0, 1].set_ylabel('Average Review Impact Score', color='darkred', size=15)

df_clean['year'] = df_clean['last_review'].dt.year
average_price_yearly = df_clean.groupby(['year',
'seasonal_booking'])['price'].mean().unstack()
average_price_yearly.plot(ax=axs[1, 0], marker='o')
axs[1, 0].set_title('Average Price by Room Type Over Time', fontname='serif',
color='blue', size=15)
axs[1, 0].set_xlabel('Year', color='darkred', size=15)
axs[1, 0].set_ylabel('Average Price ($)', color='darkred', size=15)
axs[1, 0].legend(title='Room Type')
axs[1, 0].set_xlim(left=df_clean['year'].min(), right=df_clean['year'].max())

top_neighbourhoods_occupancy =
df_clean.groupby('neighbourhood')['occupancy_rate'].mean().nlargest(5)
sns.barplot(x=top_neighbourhoods_occupancy.index,
y=top_neighbourhoods_occupancy.values, ax=axs[1, 1], palette = 'Set2')
axs[1, 1].set_title('Top 5 Neighbourhoods by Average Occupancy Rate',
fontname='serif', color='blue', size=15)
axs[1, 1].set_xlabel('Neighbourhood', color='darkred', size=15,)
axs[1, 1].set_ylabel('Average Occupancy Rate', color='darkred', size=15)
axs[1, 1].tick_params(axis='x', rotation=0)
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.suptitle('Trend Analysis', fontsize=20,fontname='serif', color='blue')
plt.show()

#%% Lm or reg plot with scatter representation and regression line

df_clean['reviews_per_month'] = df_clean['reviews_per_month'].replace(0,
np.nan)
df_clean['review_density'] = df_clean['number_of_reviews'] /
df_clean['reviews_per_month']
df_clean = df_clean.dropna(subset=['review_density'])

sns.lmplot(x='number_of_reviews', y='review_density', data=df_clean,
           height=6, aspect=1.5, scatter_kws={'alpha':0.6},
line_kws={'color':'red'})

plt.title('Linear model plot(lm) with Regression Line',fontname='serif',
color='blue', size=15)
plt.xlabel('Number of Reviews',color='darkred', size=15, labelpad=10)
plt.ylabel('Review Density',color='darkred', size=15, labelpad=10)
plt.tight_layout()
plt.grid(True)
plt.show()

correlation = df_clean[['number_of_reviews',
'review_density']].corr().iloc[0, 1]

table = PrettyTable(title="Correlation Between Number of Reviews and Review
Density")
table.field_names = ["Statistic", "Value"]
table.add_row(["Correlation Coefficient", f"{correlation:.4f}"])
table.align = "l"
```

```
table.align["Value"] = "r"

print(table)


#%% Boxen Plot

neighbourhood_names = ["Tribeca", "Sea Gate", "Riverdale", "Prince's Bay",
"Battery Park City",
      "Flatiron District", "Randall Manor", "NoHo", "SoHo", "Midtown"]

df_neighs = df_clean[df_clean['neighbourhood'].isin(neighbourhood_names)]
plt.figure(figsize=(10, 6))
sns.catplot(x="price", y="neighbourhood", hue = 'neighbourhood',
kind="boxen", data=df_neighs,
            palette="tab10", linewidth=1.5, alpha=0.6, height=8, aspect=2)

plt.xlabel('Price',color='darkred', size=15, labelpad=10)
plt.ylabel('')
plt.yticks(color='darkred', size=15)
plt.title('Price behavior in relation to neighbourhoods', fontsize=23,
loc='center',fontname='serif', color='blue')
plt.tight_layout()
plt.grid(True)
plt.show()

table = PrettyTable(title="Price Statistics by neighbourhood")
table.field_names = ["neighbourhood", "Min Price", "25% Quantile", "Median
Price", "75% Quantile", "Max Price"]

for neighbourhood in neighbourhood_names:
    prices = df_neighs[df_neighs['neighbourhood'] == neighbourhood]['price']
    min_price = prices.min()
    q25 = prices.quantile(0.25)
    median_price = prices.median()
    q75 = prices.quantile(0.75)
    max_price = prices.max()

    table.add_row(
        [neighbourhood, f"${min_price:.2f}", f"${q25:.2f}",
f"${median_price:.2f}", f"${q75:.2f}", f"${max_price:.2f}"])

table.align = "l"
table.align["Min Price"] = "r"
table.align["25% Quantile"] = "r"
table.align["Median Price"] = "r"
table.align["75% Quantile"] = "r"
table.align["Max Price"] = "r"

print(table)

#%% Area Plot

occupancy_by_season =
df_clean.groupby('seasonal_booking')['occupancy_rate'].mean()

plt.figure(figsize=(12, 6))
```

```
occupancy_by_season.plot(kind='area', alpha=0.5, color='green')
plt.title('Average Occupancy Rate by Season',fontname='serif', color='blue',
size=15)
plt.xlabel('Season',color='darkred', size=15, labelpad=10)
plt.ylabel('Average Occupancy Rate',color='darkred', size=15, labelpad=10)
plt.grid(True)
plt.show()

table = PrettyTable(title="Seasonal Occupancy Rates")

table.field_names = ["Season", "Average Occupancy Rate"]

for season, rate in occupancy_by_season.items():
    table.add_row([season, f"{rate:.2f}"])

table.align = "l"

print(table)



#%% VIOLIN PLOT

sns.set(style="whitegrid")

plt.figure(figsize=(12, 6))
palette_colors = sns.color_palette("husl")

ax = sns.violinplot(x='room_type', y='review_rate_number', data=df_clean,
palette=palette_colors, inner='box')

plt.title('Review Rate Number Distribution by Room Type', fontname='serif',
color='blue', size=20, pad=15)
plt.xlabel('Room Type', fontname='serif', color='darkred', size=20,
labelpad=15)
plt.ylabel('Review Rate Number', fontname='serif', color='darkred', size=20,
labelpad=15)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.tight_layout()
plt.grid(True)
plt.show()

grouped_stats =
df_clean.groupby('room_type')['review_rate_number'].describe()
table = PrettyTable(title="Review Rate Number Distribution by Room Type")

table.field_names = ["Room Type", "Count", "Mean", "Std", "Min", "25%",
"50%", "75%", "Max"]

for room_type, stats in grouped_stats.iterrows():
    table.add_row([
        room_type,
        f"{stats['count']:.0f}",
        f"{stats['mean']:.2f}",
        f"{stats['std']:.2f}",
        f"{stats['min']:.2f}",
```

```
        f"{stats['25%']:.2f}",
        f"{stats['50%']:.2f}",
        f"{stats['75%']:.2f}",
        f"{stats['max']:.2f}"
    ])

table.align = "r"
table.align["Room Type"] = "l"

print(table)

#%% Booking Features Distribution (Third Sub Plot

fig, axs = plt.subplots(1, 3, figsize=(18, 6))
custom_palette = sns.color_palette("Set2")
seasonal_booking_counts = df_clean['seasonal_booking'].value_counts()
explode = [0,0,0,0.08]

plt.subplot(131)
plt.pie(seasonal_booking_counts, labels=seasonal_booking_counts.index,
autopct='%1.2f%%', startangle=140, colors=custom_palette, explode=explode)
axs[0].set_title('Seasonal Booking Distribution',fontname='serif',
color='blue', size=15)

plt.subplot(132)
instant_bookable_counts =
df_clean['instant_bookable'].value_counts(normalize=True)
plt.pie(instant_bookable_counts, labels=instant_bookable_counts.index,
autopct='%1.2f%%', startangle=140, colors=custom_palette)
axs[1].set_title('Instant Bookable Distribution',fontname='serif',
color='blue', size=15)

plt.subplot(133)
cancellation_policy_proportions =
df_clean['cancellation_policy'].value_counts(normalize=True)
plt.pie(cancellation_policy_proportions,
labels=cancellation_policy_proportions.index, autopct='%1.2f%%',
startangle=140, colors=custom_palette)
axs[2].set_title('Cancellation Policy Distribution',fontname='serif',
color='blue', size=15)

axs[0].legend(bbox_to_anchor=(1.05, 1), loc='upper left')
axs[1].legend(bbox_to_anchor=(1.05, 1), loc='upper left')
axs[2].legend(bbox_to_anchor=(1.05, 1), loc='upper left')

plt.suptitle('Booking Features Distribution',fontname='serif', color='blue',
size=25)
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()

#%% Joint plot with KDE and Scatter representation

np.random.seed(42)
df_clean['host_response_rate'] = np.random.uniform(50, 100, len(df_clean))
df_clean['listing_popularity'] = np.log1p(df_clean['number_of_reviews'] *
df_clean['reviews_per_month'])
```

```
sampled_data = df_clean.sample(n=1000, random_state=42)
joint_kde_scatter_plot = sns.jointplot(
    x='host_response_rate',
    y='listing_popularity',
    data=sampled_data,
    kind='kde',
    color='purple',
    space=0,
    fill=True
)
joint_kde_scatter_plot.ax_joint.scatter(sampled_data['host_response_rate'],
sampled_data['listing_popularity'], color='purple', alpha=0.6)

joint_kde_scatter_plot.set_axis_labels('Host Response Rate (%)', 'Listing
Popularity', fontsize=12,color='darkred', labelpad=10)
plt.subplots_adjust(top=0.9)
plt.suptitle('Joint Plot with KDE and Scatterplot', fontsize=15,
fontname='serif', color='blue')
plt.tight_layout()
plt.show()

host_response_rate_mean = sampled_data['host_response_rate'].mean()
host_response_rate_std = sampled_data['host_response_rate'].std()
listing_popularity_mean = sampled_data['listing_popularity'].mean()
listing_popularity_std = sampled_data['listing_popularity'].std()
host_response_rate_median = sampled_data['host_response_rate'].median()
listing_popularity_median = sampled_data['listing_popularity'].median()


table = PrettyTable(title = "Statistics of Host Response Rate and Listing
Popularity")
table.field_names = ["Statistics", "Host Response Rate", "Listing
Popularity"]
table.add_row(["Mean",
f"{host_response_rate_mean:.2f}",f"{listing_popularity_mean:.2f}"])
table.add_row(["Median", f"{host_response_rate_median:.2f}",
f"{listing_popularity_median:.2f}"])
table.add_row(["Standard Deviation",
f"{host_response_rate_std:.2f}",f"{listing_popularity_std:.2f}"])
# Print pretty table
print(table)

#%% Rug Plot

sns.set(style="whitegrid")
plt.figure(figsize=(10, 4))
sns.rugplot(df_clean['review_impact_score'], height=0.5, color='red')

plt.title('Rug Plot of Review Impact Score', fontsize=16, fontname='serif',
color='blue')
plt.xlabel('Review Impact Score', color='darkred', size=15, labelpad=10)
plt.ylabel('Density',color='darkred', size=15, labelpad=10)
plt.tight_layout()
plt.grid(True)
plt.show()

stats = df_clean['review_impact_score'].describe()
```

```
table = PrettyTable(title="Statistics of Review Impact Scores")
table.field_names = ["Statistic", "Value"]

statistics = ['count', 'mean', 'std', 'min', '25%', '50%', '75%', 'max']
for stat in statistics:
    table.add_row([stat.capitalize(), f"{stats[stat]:.2f}"])

table.align = "l"
table.align["Value"] = "r"

print(table)

#%% 3D plot

# 3D plot
df_clean.replace([np.inf, -np.inf], np.nan, inplace=True)
df_clean.dropna(subset=['Host Activity Score', 'number_of_reviews'],
inplace=True)

df_clean['log_number_of_reviews'] = np.log1p(df_clean['number_of_reviews'])

fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection='3d')

scat = ax.scatter(df_clean['Host Activity Score'],
                  df_clean['log_number_of_reviews'],
                  df_clean['price'],
                  c=df_clean['price'],
                  cmap='plasma')

cbar = fig.colorbar(scat, shrink=0.5, aspect=5)
cbar.set_label('Price')

ax.set_xlabel('Host Activity Score',color='darkred', size=15, labelpad=10)
ax.set_ylabel('Number of Reviews',color='darkred', size=15, labelpad=10)
ax.set_zlabel('Price',color='darkred', size=15, labelpad=10)

ax.set_title('3D Plot of Price vs. Host Activity Score and Number of
Reviews',fontsize=16, fontname='serif', color='blue')
plt.tight_layout()
plt.grid(True)
plt.show()

stats_price = df_clean['price'].describe()
stats_host_activity = df_clean['Host Activity Score'].describe()
stats_log_reviews = df_clean['log_number_of_reviews'].describe()

table = PrettyTable(title="Summary Statistics for Key Metrics")

table.field_names = ["Metric", "Count", "Mean", "Std", "Min", "25%", "50%
(Median)", "75%", "Max"]

table.add_row(["Price", f"{stats_price['count']:.0f}",
f"${stats_price['mean']:.2f}", f"${stats_price['std']:.2f}",
               f"${stats_price['min']:.2f}", f"${stats_price['25%']:.2f}",
f"${stats_price['50%']:.2f}",
```

```
                f"${stats_price['75%']:.2f}", f"${stats_price['max']:.2f}"])
table.add_row(["Host Activity Score", f"{stats_host_activity['count']:.0f}",
f"{stats_host_activity['mean']:.2f}",
                f"{stats_host_activity['std']:.2f}",
f"{stats_host_activity['min']:.2f}", f"{stats_host_activity['25%']:.2f}",
                f"{stats_host_activity['50%']:.2f}",
f"{stats_host_activity['75%']:.2f}", f"{stats_host_activity['max']:.2f}"])
table.add_row(["Number of Reviews", f"{stats_log_reviews['count']:.0f}",
f"{stats_log_reviews['mean']:.2f}",
                f"{stats_log_reviews['std']:.2f}",
f"{stats_log_reviews['min']:.2f}", f"{stats_log_reviews['25%']:.2f}",
                f"{stats_log_reviews['50%']:.2f}",
f"{stats_log_reviews['75%']:.2f}", f"{stats_log_reviews['max']:.2f}"])

table.align = "l"
table.align["Count"] = "r"
table.align["Mean"] = "r"
table.align["Std"] = "r"
table.align["Min"] = "r"
table.align["25%"] = "r"
table.align["50% (Median)"] = "r"
table.align["75%"] = "r"
table.align["Max"] = "r"

print(table)



#%% Contour Plot

np.random.seed(42)
sample_size = 1000
x = np.random.uniform(1, 1000, sample_size)  # For Price
y = np.random.uniform(1, 100, sample_size)   # For Number of Reviews
z = np.random.uniform(0, 5, sample_size)     # For Value Score

xi, yi = np.linspace(x.min(), x.max(), 100), np.linspace(y.min(), y.max(),
100)
xi, yi = np.meshgrid(xi, yi)

kde = gaussian_kde(np.vstack([x, y]))

zi = kde(np.vstack([xi.flatten(), yi.flatten()]))
zi = zi.reshape(xi.shape)

fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection='3d')

surf = ax.plot_surface(xi, yi, zi, rstride=1, cstride=1, cmap='plasma',
linewidth=0, antialiased=False, alpha = 0.5)

levels = np.linspace(zi.min(), zi.max(), 15)
ax.contourf(xi, yi, zi, levels=levels, zdir='z', offset=ax.get_zlim()[0],
cmap='plasma', alpha=0.5)

for level in levels[::5]:
    ax.contour(xi, yi, zi, levels=[level], zdir='z', offset=level,
```

```
cmap='plasma', alpha=0.5)

cbar = fig.colorbar(surf, shrink=0.5, aspect=5)
cbar.set_label('Density')

ax.set_xlabel('Price',color='darkred', size=15, labelpad=10)
ax.set_ylabel('Number of Reviews',color='darkred', size=15, labelpad=10)
ax.set_zlabel('Value Score',color='darkred', size=15)

plt.title('3D Gaussian KDE with Contours',fontsize=16, fontname='serif',
color='blue')

plt.show()

#%% Cluster map

selected_columns = [
    'occupancy_rate', 'performance_score', 'Host Activity Score',
    'review_impact_score', 'monthly_revenue'
]

df_selected = df_clean[selected_columns]

correlation_matrix = df_selected.corr()

clustermap_fig = sns.clustermap(correlation_matrix, cmap="viridis",
figsize=(16, 10), annot=True, fmt=".2f", xticklabels=True, yticklabels=True)
clustermap_fig.ax_heatmap.set_xticklabels(clustermap_fig.ax_heatmap.get_xtick
labels(), rotation=0,color='darkred', size=15)
clustermap_fig.ax_heatmap.set_yticklabels(clustermap_fig.ax_heatmap.get_ytick
labels(),color='darkred', size=15)
plt.title('Cluster Map', fontsize=16, fontname='serif', color='blue')

plt.tight_layout()
plt.show()



#%% Hexbin

np.random.seed(42)

sampled_df = df_clean.sample(n=5000, random_state=42)

max_price = sampled_df['price'].quantile(0.95)
max_reviews = sampled_df['review_density'].quantile(0.95)

filtered_df = sampled_df[(sampled_df['price'] <= max_price) &
                         (sampled_df['review_density'] <= max_reviews)]

plt.figure(figsize=(10, 6))
hb = plt.hexbin(filtered_df['price'], filtered_df['review_density'],
                gridsize=40, cmap='viridis', mincnt=1, alpha=0.6)
cb = plt.colorbar(hb)
cb.set_label('Counts in Bin')
plt.title('Hexbin Plot of Price vs. Review Density',
```

```
fontsize=16,fontname='serif', color='blue')
plt.xlabel('Price', color='darkred', size=15,)
plt.ylabel('Review Density', color='darkred', size=15)
plt.tight_layout()
plt.grid(True)
plt.show()




#%% Strip Plot

plt.figure(figsize=(12, 8))
strip_plot = sns.stripplot(
    x='price',
    y='room_type',
    hue='host_identity_verified',
    data=df_clean,
    jitter=0.25,
    palette='viridis',
    dodge=True
)
plt.title('Price Distribution by Room Type and Host Identity Verification',
fontsize=16, fontname='serif', color='blue')
plt.xlabel('Price', color='darkred', size=15, labelpad=10)
plt.ylabel('Room Type', color='darkred', size=15, labelpad=10)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.legend(title='Host Identity Verified', bbox_to_anchor=(1, 1), loc='upper
left')

plt.tight_layout()
plt.grid(True)
plt.show()

#%% Swarm Plot

sample_data = df_clean.sample(n=2500, random_state=1)
plt.figure(figsize=(10, 6))
sns.swarmplot(x='room_type', y='price', data=sample_data, hue = 'room_type',
palette = 'Set2')

y_labels = ['${}'.format(int(y)) for y in plt.gca().get_yticks()]
plt.gca().set_yticklabels(y_labels)
plt.title('Swarm Plot of Room Type vs Price',fontname='serif', color='blue',
size=15)
plt.xlabel('Room Type',color='darkred', size=15, labelpad=10)
plt.ylabel('Price',color='darkred', size=15, labelpad=10)
plt.grid(True)
plt.show()

grouped_data = sample_data.groupby('room_type')['price'].agg(['min',
'median', 'max'])

table = PrettyTable(title="Price Statistics by Room Type")
```

```
table.field_names = ["Room Type", "Minimum Price", "Median Price", "Maximum
Price"]

for room_type, row in grouped_data.iterrows():
    table.add_row([
        room_type,
        f"${row['min']:.0f}",
        f"${row['median']:.0f}",
        f"${row['max']:.0f}"
    ])

table.align = "l"
table.align["Minimum Price"] = "r"
table.align["Median Price"] = "r"
table.align["Maximum Price"] = "r"

print(table)
#%% Fourth Subplot

fig, axs = plt.subplots(2, 2, figsize=(23, 20))

countplot = sns.countplot(x="neighbourhood_group", hue =
'neighbourhood_group',data=df_clean, ax=axs[0, 0], palette="Set2")
axs[0, 0].set_title('Listing Count by Neighbourhood Group',fontsize=15,
fontname='serif', color='blue')
axs[0, 0].set_xlabel('Neighbourhood Group',color='darkred', size=15,
labelpad=10)
axs[0, 0].set_ylabel('Number of Listings',color='darkred', size=15,
labelpad=10)

for c in countplot.patches:
    axs[0, 0].text(c.get_x() + c.get_width() / 2.,
                   c.get_height(),
                   '{:1.2f}'.format(c.get_height()),
                   ha="center",
                   va="bottom",
                   fontsize=15,
                   color='black',
                   rotation=0)

sns.barplot(x='neighbourhood_group', y='price', hue='room_type',
data=df_clean, ax=axs[0, 1], palette="cool")
axs[0, 1].set_title('Average Price by Neighbourhood Group and Room
Type',fontsize=20, fontname='serif', color='blue')
axs[0, 1].set_xlabel('Neighbourhood Group',color='darkred', size=20,
labelpad=10)
axs[0, 1].set_ylabel('Average Price',color='darkred', size=20, labelpad=10)
axs[0, 1].legend(title='Room Type')

df_review_counts =
df_clean.groupby('neighbourhood_group')['number_of_reviews'].mean().reset_ind
ex()
df_review_counts = df_review_counts.sort_values('number_of_reviews',
ascending=False)
barplot = sns.barplot(x='neighbourhood_group', y='number_of_reviews',
data=df_review_counts, ax=axs[1, 0], palette="RdPu", hue =
'neighbourhood_group')
```

```
axs[1, 0].set_title('Average Review Count by Neighbourhood
Group',fontsize=20, fontname='serif', color='blue')
axs[1, 0].set_xlabel('Neighbourhood Group',color='darkred', size=20,
labelpad=10)
axs[1, 0].set_ylabel('Average Number of Reviews',color='darkred', size=20,
labelpad=10)
for p in barplot.patches:
    axs[1, 0].text(p.get_x() + p.get_width() / 2.,
                   p.get_height(),
                   '{:1.2f}'.format(p.get_height()),
                   ha="center",
                   va="bottom",
                   fontsize=15,
                   color='black',
                   rotation=0)

sns.boxenplot(y='price', x='neighbourhood_group', data=df_clean, hue =
'neighbourhood_group', ax=axs[1, 1], palette="spring")
axs[1, 1].set_title('Price Distribution by Neighbourhood Group',fontsize=20,
fontname='serif', color='blue')
axs[1, 1].set_xlabel('Neighbourhood Group',color='darkred', size=20,
labelpad=10)
axs[1, 1].set_ylabel('Price',color='darkred', size=20, labelpad=10)
medians =
df_clean.groupby('neighbourhood_group')['price'].median().sort_index()
means = df_clean.groupby('neighbourhood_group')['price'].mean().sort_index()

x_positions = range(len(medians))

for i in x_positions:
    median_y_position = 650
    mean_y_position = 580

    axs[1, 1].text(i, median_y_position, f"Median: {medians[i]:.2f}",
ha='center', va='bottom', fontsize=15, color='black')
    axs[1, 1].text(i, mean_y_position, f"Mean: {means[i]:.2f}", ha='center',
va='bottom', fontsize=15, color='darkred')
plt.tight_layout()
plt.grid(True)
tick_fontsize = 15

for row in axs:
    for ax in row:
        for label in ax.get_xticklabels():
            label.set_fontsize(tick_fontsize)
        for label in ax.get_yticklabels():
            label.set_fontsize(tick_fontsize)
plt.subplots_adjust(top=0.94)
plt.suptitle('Neighbourhood Groups Dynamics',fontsize=20, fontname='serif',
color='blue')
plt.show()
```

Python Code of Phase II (Dash App).

```
import pandas as pd
import seaborn as sns
import dash
from dash import dcc, html
from dash.dependencies import Input, Output, State
import plotly.express as px
from scipy.stats import kstest
from scipy import stats
import numpy as np
import plotly.graph_objects as go
import re
from scipy.stats import gaussian_kde
from dash.exceptions import PreventUpdate


#%% Importing the dataset and setting the style

df = pd.read_csv('Airbnb_open_data.csv', encoding='latin1')
print(df)
sns.set(style="whitegrid")

#%% Checking for Null Values

df.isnull().sum()
#%%

columns_to_drop = ['license', 'house_rules']
df_clean = df.drop(columns=columns_to_drop, errors='ignore')
df_clean.info()

#%%
df_clean.dropna(inplace = True)
print("df type:", type(df_clean))
print("df shape:", df_clean.shape)

#%%

before_duplicates = len(df_clean)
df_clean.drop_duplicates(inplace=True)
after_duplicates = len(df_clean)

print(f"Total number of records before removing duplicates:
{before_duplicates}")
print(f"Total number of records after removing duplicates:
{after_duplicates}")

#%%
df_clean.rename(columns={'availability 365': 'days_booked'}, inplace=True)
df_clean.columns = df_clean.columns.str.lower().str.replace(' ', '_')

df_clean['price'] = df_clean['price'].replace({'\$': '', ',': ''},
regex=True)
df_clean['service_fee'] = df_clean['service_fee'].replace({'\$': '', ',':
```

```
''}, regex=True)

df_clean['price'] = pd.to_numeric(df_clean['price'], errors='coerce')
df_clean['service_fee'] = pd.to_numeric(df_clean['service_fee'],
errors='coerce')

#%%
df_clean['neighbourhood_group'].unique()

#%%
corrections = {
    'brookln': 'Brooklyn',
    'manhatan': 'Manhattan'
}
df_clean['neighbourhood_group'] =
df_clean['neighbourhood_group'].replace(corrections)
print(df_clean['neighbourhood_group'].unique())

#%%

room_type_counts = df_clean ['room_type'].value_counts()
print("Count of Various Room Types Available:")
print(room_type_counts)

#%%

strict_cancellation =
df_clean.groupby('room_type')['cancellation_policy'].value_counts().groupby(l
evel=0).head(1)
print("Room type with the most strict cancellation policy:")
print(strict_cancellation)

#%%
avg_price_neighbourhood =
df_clean.groupby('neighbourhood_group')['price'].mean().sort_values(ascending
=False)
print("Average price per neighbourhood group:")
print(avg_price_neighbourhood)

#%% Feature Engineering

average_price_per_room_type = df_clean.groupby('room_type')['price'].mean()
avg_price_neighbourhood = df_clean.groupby('neighbourhood')['price'].mean()

sorted_neighbourhoods_expensive =
avg_price_neighbourhood.sort_values(ascending=False)

top_10_expensive_neighbourhoods = sorted_neighbourhoods_expensive.head(10)

top_10_cheapest_neighbourhoods = sorted_neighbourhoods_expensive.tail(10)

df_clean_expensive_neighbourhoods =
top_10_expensive_neighbourhoods.reset_index()
df_clean_cheapest_neighbourhoods =
top_10_cheapest_neighbourhoods.reset_index()

print("Top 10 expensive neighbourhoods:")
```

```
for index, row in df_clean_expensive_neighbourhoods.iterrows():
    print(f"{row['neighbourhood']}: ${row['price']:.2f}")

print("\nTop 10 cheapest neighbourhoods:")
for index, row in df_clean_cheapest_neighbourhoods.iterrows():
    print(f"{row['neighbourhood']}: ${row['price']:.2f}")

df_clean['last_review'] = pd.to_datetime(df_clean['last_review'],
errors='coerce')

# 1. Monthly Revenue
df_clean['monthly_revenue'] = df_clean['price'] * df_clean['days_booked']

# 2. Occupancy Rate
df_clean['occupancy_rate'] = df_clean['days_booked'] / 30

# 3. Average Daily Rate (ADR)
df_clean['average_daily_rate'] = df_clean['monthly_revenue'] /
df_clean['days_booked']

# 4. Review Impact Score
df_clean['review_impact_score'] = df_clean['number_of_reviews'] *
df_clean['review_rate_number']

# 5. Seasonal Bookings (simplified to quarters)
df_clean['seasonal_booking'] = pd.cut(df_clean['last_review'].dt.month,
                                      bins=[0, 3, 6, 9, 12],
                                      labels=['Winter', 'Spring', 'Summer',
'Autumn'],
                                      right=False)

# 6. Host Efficiency
df_clean['host_efficiency'] = df_clean['calculated_host_listings_count'] /
df_clean['number_of_reviews']
df_clean['host_efficiency'] = df_clean['host_efficiency'].replace(np.inf, 0)

# 7. Host Activity Score
df_clean['Host Activity Score'] = df_clean['calculated_host_listings_count']
* df_clean['review_rate_number']

df_clean['reviews_frequency'] = np.log1p(df_clean['reviews_per_month'])

df_clean['review_impact_trend'] =
df_clean.groupby(df_clean['last_review'].dt.year)['review_impact_score'].tran
sform('mean')
df_clean['year'] = df_clean['last_review'].dt.year


#
df_clean['reviews_per_month'] = df_clean['reviews_per_month'].replace(0,
np.nan)
df_clean['review_density'] = df_clean['number_of_reviews'] /
df_clean['reviews_per_month']

df_clean['host_response_rate'] = np.random.uniform(50, 100, len(df_clean))
df_clean['listing_popularity'] = np.log1p(df_clean['number_of_reviews'] *
df_clean['reviews_per_month'])
```

```
#%%

room_type = df_clean['room_type'].unique()

print(df_clean.info())


#%%

from scipy.stats import zscore

numerical_columns = df.select_dtypes(include='number')

z_scores = numerical_columns.apply(zscore)

z_outliers = df[(z_scores > 3).any(axis=1)]

Q1 = numerical_columns.quantile(0.25)
Q3 = numerical_columns.quantile(0.75)
IQR = Q3 - Q1

IQR_outliers = df[((numerical_columns < (Q1 - 1.5 * IQR)) |
(numerical_columns > (Q3 + 1.5 * IQR))).any(axis=1)]

#%%
outliers_count = IQR_outliers.select_dtypes(include=['number']).apply(lambda
x: x.isna().sum(), axis=0)

top_outliers_columns = outliers_count.sort_values(ascending=False).head(5)

print("Top 5 columns with outliers:")
print(top_outliers_columns)

#%%

def perform_normality_test(data):
    statistic, p_value = kstest(data, 'norm')
    return {
        'statistic': statistic,
        'p_value': p_value
    }

def remove_outliers(data, z_thresh=3):
    # Assuming normal distribution, we'll remove data 3 standard deviations
from the mean
    z_scores = stats.zscore(data)
    return data[(z_scores < z_thresh) & (z_scores > -z_thresh)]

def identify_outliers(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    return ((column < (Q1 - 1.5 * IQR)) | (column > (Q3 + 1.5 * IQR)))
```

```
#%%

# ############################
# # PHASE 1
# ############################
external_stylesheets = ['https://codepen.io/chriddyp/pen/bWLwgP.css']


# ############################
# # PHASE 2 App
# ############################

app = dash.Dash('FTP', external_stylesheets=external_stylesheets)

# ############################
# # PHASE 3 Layout
# ############################

app.layout = html.Div([
    html.Div([
        html.Img(src='assets/Airbnb_logo.jpeg', style={'width': '50px',
'height': '50px', 'vertical-align': 'middle'}),
        html.H1("Airbnb",
                style={'display': 'inline-block', 'margin-left':
'10px','margin-top': '20px', 'font-family': 'Times New Roman', 'font-weight':
'bold', 'color':'#FF464B', 'vertical-align': 'middle', 'font-size': '50px'}),
    ], style={'text-align': 'center', 'margin-bottom': '20px'}),

    dcc.Tabs(id='tabs', value='tab-all',
             children=[
                 dcc.Tab(label='Stays', value='Stays', style={'border':
'none'}),
                 dcc.Tab(label='Normality Tests & Outlier Detection',
value='NormOut', style={'border': 'none'}),
                 dcc.Tab(label= 'Statistical Analysis', value='Stats', style
= {'border':'none'}),
                 dcc.Tab(label='Accommodation Trends', value='Trend',
style={'border': 'none'}),
                 dcc.Tab(label='Sign up', value='Sign up', style = {'border':
'none'}),
             ],
             style={'border-bottom': '2px solid #FF464B', 'font-family':
'Times New Roman', 'font-size': '18px', 'color': '#333', 'background-color':
'#f5f5f5', 'padding': '10px', 'margin-bottom': '20px'}
             ),

    html.Div(id='tabs-content')
])

# ############################
# # Stays layout
# ############################

df_clean['Details'] = df_clean['name'] + ' , ' + df_clean['neighbourhood'] +
'<br>Room Type:' + df_clean['room_type'] + '<br>Price: $' +
df_clean['price'].astype(str)
stays_layout = html.Div([
```

```
    html.H1("Explore Airbnb Listings in New York!", style={'textAlign':
'center', 'color': '#FF464B','font-family': 'Times New Roman'}),

    html.Div([
        html.P(
            "Welcome to our Airbnb exploration tool! Use the interactive map
below to discover available listings in New York City."),
        html.P("Adjust the price range and select room types and neighborhood
groups to refine your search."),
    ], style={'margin': '20px', 'padding': '20px', 'background-color':
'#f5f5f5', 'border-radius': '10px'}),

    dcc.Graph(id='airbnb_map', style={'height': '800px', 'width': '100%'}),
    html.Br(),
    html.Div([
        html.H5('Please select a price range:', style={'font-weight':
'bold','font-family': 'Times New Roman'}),
        dcc.RangeSlider(
            id='price-slider',
            min=df_clean['price'].min(),
            max=df_clean['price'].max(),
            step=10,
            marks={i: '${}'.format(i) for i in
range(int(df_clean['price'].min()), int(df_clean['price'].max()), 100)},
            value=[df_clean['price'].min(), df_clean['price'].max()]
        )
    ], style={'padding': '20px 60px','border': '1px solid #ddd', 'border-
radius': '5px',
              'box-shadow': '2px 2px 2px lightgrey','background-color':
'#e9ecef'}),
    html.Br(),
    html.Div([
        html.H5('Please select a room type:', style={'font-weight': 'bold',
'font-family': 'Times New Roman'}),
        dcc.RadioItems(
            id='room-type-radio',
            options=[{'label': rt, 'value': rt} for rt in room_type],
            value=room_type[0],
            labelStyle={'display': 'block', 'margin': '10px', 'font-family':
'Times New Roman'},
            style={ 'font-size': '16px'}
        ),
    ], style={'padding': '20px 60px', 'border': '1px solid #ddd', 'border-
radius': '5px',
              'box-shadow': '2px 2px 2px lightgrey','background-color':
'#e9ecef'}),

    html.Br(),
    html.Div([
        html.H5('Please select a neighbourhood group:', style={'font-weight':
'bold','font-family': 'Times New Roman'}),
        dcc.Dropdown(
            id='neighbourhood-dropdown',
            options=[
                {'label': 'Brooklyn', 'value': 'Brooklyn'},
                {'label': 'Staten Island', 'value': 'Staten Island'},
                {'label': 'Bronx', 'value': 'Bronx'},
```

```
                {'label': 'Queens', 'value': 'Queens'},
                {'label': 'Manhattan', 'value': 'Manhattan'}
            ] ,style={'font-family': 'Times New Roman'},
            multi=True,

            placeholder='Select neighbourhood group...'
        )
    ], style={'padding': '20px 60px','font-family': 'Times New Roman',
'background-color': '#fff', 'border-radius': '10px', 'box-shadow': '0 4px 8px
rgba(0,0,0,0.1)'}),

    html.Div(id='neighbourhood-details', style={'padding': '20px
60px','background-color': '#f5f5f5', 'border-radius': '10px'})
])


@app.callback(
    Output('airbnb_map', 'figure'),
    [Input('price-slider', 'value'),
     Input('room-type-radio', 'value')]
)

def update_map(price_range, selected_room_type):
    filtered_df = df_clean[(df_clean['price'] >= price_range[0]) &
(df_clean['price'] <= price_range[1]) & (df_clean['room_type'] ==
selected_room_type)]
    fig = px.scatter_mapbox(
        filtered_df,
        lat='lat',
        lon='long',
        color='neighbourhood_group',
        color_continuous_scale=px.colors.cyclical.IceFire,
        size_max=20,
        zoom=10,
        hover_name='name',
        text='Details',
        center={"lat": df_clean['lat'].mean(), "lon":
df_clean['long'].mean()}
    ).update_layout(
        legend_title_text='Neighbourhood Group',
        mapbox_style="open-street-map",
        margin={"r":0, "t":0, "l":0, "b":0},
        transition_duration=500
    )
    return fig

@app.callback(
    Output('neighbourhood-details', 'children'),
    [Input('neighbourhood-dropdown', 'value')]
)

def update_neighbourhood_details(selected_neighbourhoods):
    if not selected_neighbourhoods:
        return html.Div('No neighbourhood selected', style={'font-family':
'Times New Roman'})

    details = []
```

```
    for group in selected_neighbourhoods:
        group_df = df_clean[df_clean['neighbourhood_group'] == group]
        if not group_df.empty:
            representative_listing = group_df.iloc[0]
            image_filename = group.lower().replace(' ', '_') + '.jpg'
            image_path = f"/assets/{image_filename}"

            details_text = representative_listing['Details']
            if 'Price:' in details_text:
                details_text = details_text.split('Price:')[0].strip()

            details_parts = details_text.split('<br>')
            details_elements = [html.P(part) for part in details_parts if
part]

            card_content = html.Div([
                html.Img(src=image_path, style={'width': '100%', 'max-width':
'400px', 'height': 'auto'}),
                html.H4(f"{representative_listing['name']}"),
                *details_elements,
                html.P(f"Neighbourhood:
{representative_listing['neighbourhood']}"),
                html.P(f"Price: ${representative_listing['price']}"),
            ], style={'border': '1px solid #ccc', 'padding': '10px', 'margin-
top': '10px', 'margin-bottom': '20px', 'font-family': 'Times New Roman'})

            details.append(card_content)

    return details


# ############################
# # Normality Tests & Outlier Detection
# ############################

info_layout = html.Div([
    html.H1('Normality Test and Outlier Analysis',style={'fontFamily': 'Times
New Roman'}),
    html.Div([
        html.H3('Select a feature for normality testing:',
style={'fontFamily': 'Times New Roman'}),
        dcc.Dropdown(
            id='feature-dropdown',
            options=[{'label': 'days_booked', 'value': 'days_booked'}],
            placeholder='Please select a column',
            style={'fontFamily': 'Times New Roman'}
        ),
        html.H3('Select a normality test:', style={'fontFamily': 'Times New
Roman'}),
        dcc.Dropdown(
            id='test-dropdown',
            options=[
                {'label': 'Kolmogorov-Smirnov Test', 'value': 'ks'},
                {'label': "D'Agostino's K-squared Test", 'value':
'dagostino'}
            ],
            placeholder='Please select a test',
```

```
            value='ks',
            style={'fontFamily': 'Times New Roman'}
        ),
        dcc.Loading(
            id="loading-histogram",
            type="default",
            children=[
                dcc.Graph(id='histogram-plot'),
                html.A('Download Histogram Data',
                       id='download-histogram',
                       download="histogram.csv",
                       href="",
                       target="_blank",
                       style={'display': 'inline-block',
                              'padding': '10px 20px',
                              'background-color': '#007bff',
                              'color': '#fff',
                              'text-decoration': 'none',
                              'border': '2px solid #007bff',
                              'border-radius': '5px'}
                       )
            ]
        ),

        dcc.Loading(
            id="loading-qq",
            type="default",
            children=[
                dcc.Graph(id='qq-plot'),
                html.A('Download QQ Plot Data',
                       id='download-qq',
                       download="qqplot.csv",
                       href="",
                       target="_blank",
                       style={'display': 'inline-block',
                              'padding': '10px 20px',
                              'background-color': '#007bff',
                              'color': '#fff',
                              'text-decoration': 'none',
                              'border': '2px solid #007bff',
                              'border-radius': '5px'}
                       )
            ]
        ),
        html.Br(),
        html.Div(id='p-value-result', style={'fontFamily': 'Times New
Roman'}),
        html.H3('Select a column for outlier analysis:', style={'fontFamily':
'Times New Roman'}),
        dcc.Dropdown(
            id='column-dropdown',
            options=[{'label': col, 'value': col} for col in ['reviews per
month', 'availability 365']],
            placeholder='Please select a column',
            style={'fontFamily': 'Times New Roman'}
        ),
        html.H3('Select action:', style={'fontFamily': 'Times New Roman'}),
```

```
        dcc.RadioItems(
            id='action-selection',
            options=[
                {'label': 'Show Outliers', 'value': 'show-outliers'},
                {'label': 'Remove Outliers', 'value': 'remove-outliers'}
            ],
            value='show-outliers',
            style={'fontFamily': 'Times New Roman'}
        ),
        dcc.Loading(
            id="loading-action-plot",
            type="default",
            children=[
                dcc.Graph(id='action-plot'),
                html.A('Download Outlier Data',
                       id='download-action-plot',
                       download="outlier.csv",
                       href="",
                       target="_blank",
                       style={'display': 'inline-block',
                              'padding': '10px 20px',
                              'background-color': '#007bff',
                              'color': '#fff',
                              'text-decoration': 'none',
                              'border': '2px solid #007bff',
                              'border-radius': '5px'}
                       )
            ]
        ),
    ])
])


@app.callback(
    [
        Output('histogram-plot', 'figure'),
        Output('qq-plot', 'figure'),
        Output('p-value-result', 'children'),
        Output('download-histogram', 'href'),
        Output('download-qq', 'href'),
    ],
    [
        Input('feature-dropdown', 'value'),
        Input('test-dropdown', 'value'),
    ]
)
def update_normality_plots(feature, test):
    if feature is None:
        return {}, {}, 'Please select a feature for analysis.', "", ""

    feature_data = df_clean[feature]

    hist_figure = go.Figure(data=[go.Histogram(x=feature_data)])
    hist_figure.update_layout(title='Histogram of ' + feature)
    hist_figure.update_xaxes(title=feature)
    hist_figure.update_xaxes(title=feature, range=[0, 500])
    hist_figure.update_yaxes(title='Count')
```

```
    qq_data = stats.probplot(feature_data, dist="norm")
    qq_figure = go.Figure()
    qq_figure.add_trace(go.Scatter(x=qq_data[0][0], y=qq_data[0][1],
mode='markers', name='Data'))
    qq_figure.add_trace(
        go.Scatter(x=qq_data[0][0], y=qq_data[1][0] * qq_data[0][0] +
qq_data[1][1], mode='lines', name='Fit'))
    qq_figure.update_layout(title='QQ Plot of ' + feature)

    p_value_div = normality_test_results(feature_data, test)

    return hist_figure, qq_figure, p_value_div,
f"/download/{feature}_histogram_data.csv",
f"/download/{feature}_qqplot_data.csv"


@app.callback(
    [
        Output('action-plot', 'figure'),
        Output('download-action-plot', 'href')
    ],
    [
        Input('column-dropdown', 'value'),
        Input('action-selection', 'value')
    ]
)
def update_outlier_analysis(column, action):
    if column is None:
        return {}, ""

    column_data = df[column]

    action_figure = go.Figure()
    if action == 'show-outliers':
        action_figure.add_trace(go.Box(y=column_data, name=column))
        action_figure.update_layout(title=f'{column} with Outliers')
    elif action == 'remove-outliers':
        outliers = identify_outliers(column_data)
        filtered_data = column_data[~outliers]
        action_figure.add_trace(go.Box(y=filtered_data, name=column))
        action_figure.update_layout(title=f'{column} without Outliers')

    action_figure.update_yaxes(title=column)
    return action_figure, f"/download/{column}_outlier_data.csv"


def normality_test_results(data, test):
    try:
        if test == 'ks':
            ks_stat, ks_p_value = stats.kstest(data, 'norm',
args=(data.mean(), data.std()))
            normality_result = "normally distributed" if ks_p_value >= 0.05
else "not normally distributed"
            return html.Div([
                html.P(f'KS Statistic: {ks_stat:.4f}'),
```

```
                html.P(f'p-value: {ks_p_value:.4f}'),
                html.P(f'The data is {normality_result} at a 5% significance
level.')
            ])
        elif test == 'dagostino':
            dagostino_stat, dagostino_p_value = stats.normaltest(data)
            normality_result = "normally distributed" if dagostino_p_value >=
0.05 else "not normally distributed"
            return html.Div([
                html.P(f"D'Agostino's K-squared Statistic:
{dagostino_stat:.4f}"),
                html.P(f'p-value: {dagostino_p_value:.4f}'),
                html.P(f'The data is {normality_result} at a 5% significance
level.')
            ])
    except Exception as e:
        print(f"Error during test: {e}")
        return html.Div(f"Error during test: {str(e)}")


# ################################
# # Stats Layout
# ################################

columns_for_kde = ['Host Activity Score', 'occupancy_rate',
'number_of_reviews', 'listing_popularity']

stats_layout = html.Div([
    html.H1('Kernel Density Estimate Analysis', style={'textAlign':
'center'}),
    dcc.Dropdown(
        id='column-dropdown',
        options=[{'label': col, 'value': col} for col in columns_for_kde],
        multi=True,
        placeholder='Please select a column',
    ),
    dcc.Loading(
        id="loading-kde-graph",
        type="default",
        children=[
            dcc.Graph(id='kde-graph'),
        ]
    ),
    html.Div(id='column-stats'),

])


@app.callback(
    [Output('kde-graph', 'figure'),
     Output('column-stats', 'children')],
    [Input('column-dropdown', 'value')]
)
def update_kde_and_stats(selected_columns):
    if not selected_columns:
        raise PreventUpdate
```

```
    fig = go.Figure()
    column_stats_divs = []

    for selected_column in selected_columns:
        filtered_data = df_clean[selected_column].dropna()

        kde = gaussian_kde(filtered_data)
        x_d = np.linspace(filtered_data.min(), filtered_data.max(), 1000)
        y_d = kde(x_d)

        fig.add_trace(go.Scatter(x=x_d, y=y_d, fill='tozeroy', mode='lines',
name=selected_column))

        fig.update_layout(
            title=f'KDE Plot for Selected Columns',
            xaxis_title='Values',
            yaxis_title='Density',
            template='plotly_white',
        )

        if x_d.min() < 0:
            fig.update_xaxes(range=[0, x_d.max()])

        column_stats = filtered_data.describe().to_frame()
        column_stats_div = html.Div([
            html.H4(f'Descriptive Statistics for {selected_column}:'),
            dcc.Markdown('```' + column_stats.to_string() + '')
        ], style={'fontSize': 18, 'font-family': 'Times New Roman'})

        column_stats_divs.append(column_stats_div)

    return fig, column_stats_divs


# ################################
# # Accomodation Trends Layout
# ################################
from plotly.subplots import make_subplots

trend_analysis_layout = html.Div(
    style={
        'fontFamily': 'Times New Roman',
        'border': '2px solid #e6e6e6',
        'boxShadow': '4px 4px 8px #cccccc',
        'padding': '20px',
        'margin': '10px'
    },
    children=[
        html.H1(
            'Neighbourhood Group Analysis',
            style={'textAlign': 'left'}
        ),
        html.H3('Please select an option:'),
        dcc.Checklist(
            id='neighbourhood-checklist',
            options=[
                {'label': 'Brooklyn', 'value': 'Brooklyn'},
```

```
                {'label': 'Manhattan', 'value': 'Manhattan'},
                {'label': 'Queens', 'value': 'Queens'},
                {'label': 'Bronx', 'value': 'Bronx'},
                {'label': 'Staten Island', 'value': 'Staten Island'}
            ],
            inline=True
        ),
        dcc.Dropdown(
            id='feature-dropdown',
            options=[
                {'label': 'Number of Reviews', 'value': 'number_of_reviews'},
                {'label': 'Average Daily Rate', 'value':
'average_daily_rate'}
            ],
            placeholder='Please select a column',
            clearable=False
        ),
        dcc.Graph(id='histogram-graph'),
        html.Br(),
        html.Div([
            html.H1('Price Distribution by Room Type'),
            html.H3('Please select an option:'),
            dcc.Dropdown(
                id='dropdown-room-type',
                options=[
                    {'label': room_type, 'value': room_type} for room_type in
df_clean['room_type'].unique()
                ],multi = True,
                placeholder='Select a room type',
                clearable=False
            ),
            dcc.Graph(id='boxplot'),]),
    ]
)


@app.callback(
    Output('histogram-graph', 'figure'),
    [Input('neighbourhood-checklist', 'value'),
     Input('feature-dropdown', 'value')]
)
def update_histogram(neighbourhood_selection, selected_feature):
    if neighbourhood_selection is None or not neighbourhood_selection or
selected_feature is None:
        return {}

    sorted_df = df_clean.groupby(['neighbourhood_group',
'room_type'])[selected_feature].mean().reset_index()

    order =
sorted_df.groupby('neighbourhood_group')[selected_feature].mean().sort_values
(ascending=False).index

    num_plots = len(neighbourhood_selection)
    fig = make_subplots(rows=1, cols=num_plots,
subplot_titles=neighbourhood_selection)
```

```
    for i, neighbourhood in enumerate(neighbourhood_selection, start=1):
        filtered_df = sorted_df[sorted_df['neighbourhood_group'] ==
neighbourhood]
        fig.add_trace(
            go.Bar(x=filtered_df['room_type'],
y=filtered_df[selected_feature], name=neighbourhood),
            row=1, col=i
        )

    fig.update_layout(
        title_text=f'Distribution of {selected_feature} by Room Type for
Selected Neighbourhood Group',
        showlegend=False,
        bargap=0.2
    )

    for i, neighbourhood in enumerate(order, start=1):
        fig.update_xaxes(title_text=neighbourhood, row=1, col=i)
        fig.update_yaxes(title_text='Count')

    return fig
@app.callback(
    Output('boxplot', 'figure'),
    [Input('dropdown-room-type', 'value')]
)
def update_boxplot(selected_room_types):
    if not selected_room_types:
        return {}

    fig = make_subplots(rows=1, cols=len(selected_room_types),
subplot_titles=selected_room_types)

    for i, room_type in enumerate(selected_room_types, start=1):
        filtered_data = df_clean[df_clean['room_type'] == room_type]
        fig.add_trace(go.Box(y=filtered_data['price'], name=room_type),
row=1, col=i)

    fig.update_layout(
        title='Price Distribution by Room Type',
        yaxis=dict(title='Price'),
        font=dict(family='serif')
    )

    return fig
```

```
# ############################
# # Sign up Layout
# ############################
```

```
country_codes = [
    {'label': '+1 (United States)', 'value': '+1'},
    {'label': '+91 (India)', 'value': '+91'},
    {'label': '+44 (United Kingdom)', 'value': '+44'},
    {'label': '+86 (China)', 'value': '+86'},
    {'label': '+81 (Japan)', 'value': '+81'},
    {'label': '+7 (Russia)', 'value': '+7'},
    {'label': '+55 (Brazil)', 'value': '+55'},
    {'label': '+49 (Germany)', 'value': '+49'},
    {'label': '+33 (France)', 'value': '+33'},
    {'label': '+39 (Italy)', 'value': '+39'},
    {'label': '+82 (South Korea)', 'value': '+82'},
    {'label': '+34 (Spain)', 'value': '+34'},
    {'label': '+61 (Australia)', 'value': '+61'},
    {'label': '+52 (Mexico)', 'value': '+52'},
    {'label': '+7 (Kazakhstan)', 'value': '+7'},
    {'label': '+64 (New Zealand)', 'value': '+64'},
    {'label': '+1 (Canada)', 'value': '+1'},
    {'label': '+91 (Pakistan)', 'value': '+92'},
    {'label': '+880 (Bangladesh)', 'value': '+880'},
    {'label': '+234 (Nigeria)', 'value': '+234'}
]
sign_up_layout = html.Div(
    style={
    'background-image': 'url("/assets/airbnb_bg.jpg")',
    'background-size': 'cover',
    'height': '100vh',
    'display': 'flex',
    'justify-content': 'center',
    'align-items': 'center',
}, children=[
        html.Div(id='sign-up-form', ),
        html.H1('Sign Up Form', style={'text-align': 'center','margin':
'auto', 'width': '50%', 'padding': '20px',  'font-family':'Times New
Roman'}),
        html.Div([
        html.Label('First Name',style={'color':'white'}),
        dcc.Tooltip(
            html.Label('First Name'),
            targetable='first-name-tooltip'
        ),
        dcc.Input(id='first-name', type='text', placeholder='Enter your first
name', style={'margin-bottom': '10px'}),
        html.Br(),
        html.Label('Last Name',style={'color':'white'}),
        dcc.Tooltip(
            html.Label('Last Name'),
            targetable='last-name-tooltip'
        ),
        dcc.Input(id='last-name', type='text', placeholder='Enter your last
name', style={'margin-bottom': '10px'}),

        html.Div([
            html.Label('Age',style={'color':'white'}),
            dcc.Input(id='age', type='number', min=18, placeholder='Enter
your age', style={'margin-bottom': '10px'})
```

```
        ], title='Your age must be 18 or above'),

        html.Div([
            html.Label('Gender',style={'color':'white'}),
            dcc.Dropdown(
                id='gender',
                options=[
                    {'label': 'Male', 'value': 'Male'},
                    {'label': 'Female', 'value': 'Female'},
                    {'label': 'Other', 'value': 'Other'}
                ],
                placeholder='Select your gender',
                style={'margin-bottom': '10px'}
            )
        ], title='Your gender'),

        html.Div([
            html.Label('Address',style={'color':'white'}),
            dcc.Textarea(id='address', placeholder='Enter your address',
style={'margin-bottom': '10px'})
        ], title='Your address'),

        html.Div([
            html.Label('Country Code',style={'color':'white'}),
            dcc.Dropdown(
                id='country-code',
                options=country_codes,
                placeholder='Select country code',
                style={'margin-bottom': '10px'}
            )
        ], title='Your country code'),

        html.Div([
            html.Label('Phone Number',style={'color':'white'}),
            dcc.Input(id='phone-number', type='tel', placeholder='Enter your
phone number', style={'margin-bottom': '10px'})
        ], title='Your phone number'),

        html.Div([
            html.Label('Email',style={'color':'white'}),
            dcc.Input(id='email', type='email', placeholder='Enter your
email', style={'margin-bottom': '10px'})
        ], title='Your email address'),
            html.Div(id='sign-up-form', children=[
                html.Div(id='output-message'),
                html.Button('Submit', id='submit-button',
n_clicks=0,style={'color':'white'})
            ]),
    ], style={'margin': 'auto', 'width': '50%', 'padding': '20px', 'border':
'1px solid #ccc', 'font-family':'Times New Roman', 'font-weight':'bold'}),
    html.Div(id='output-message'),
    html.Div(id='submit-confirmation', children=[
        dcc.ConfirmDialog(
            id='confirm-dialog',
            displayed=False,
            )
        ])
```

```
])

@app.callback(
    [Output('output-message', 'children'),
     Output('submit-confirmation', 'children')],
    [Input('submit-button', 'n_clicks')],
    [State('first-name', 'value'),
     State('last-name', 'value'),
     State('age', 'value'),
     State('gender', 'value'),
     State('address', 'value'),
     State('phone-number', 'value'),
     State('email', 'value'),
     State('country-code', 'value')]
)

def submit_form(n_clicks, first_name, last_name, age, gender, address,
phone_number, email, country_code):
    if not n_clicks:
        raise dash.exceptions.PreventUpdate

    error_messages = []
    error_style = {'margin': 'auto', 'width': '50%', 'padding': '10px',
                   'border': '1px solid #ccc', 'border-radius': '5px',
                   'font-family': 'Times New Roman', 'background-color':
'#e9ecef',
                   'color': 'red', 'text-align': 'center'}

    if not first_name:
        error_messages.append('First name cannot be empty.')
    elif not re.match("^[A-Za-z]+$", first_name):
        error_messages.append('First name should only contain letters.')

    if not last_name:
        error_messages.append('Last name cannot be empty.')
    elif not re.match("^[A-Za-z]+$", last_name):
        error_messages.append('Last name should only contain letters.')

    if not age:
        error_messages.append('Age cannot be empty.')
    elif not 18 <= int(age) <= 120:
        error_messages.append('You must be between 18 and 120 years old.')

    if not gender or gender not in ['Male', 'Female', 'Other']:
        error_messages.append('Please select a valid gender.')

    if not address:
        error_messages.append('Please enter your address.')

    if not phone_number:
        error_messages.append('Phone number cannot be empty.')
    elif not re.match("^[0-9]+$", phone_number):
        error_messages.append('Phone number should only contain numbers.')

    if not email:
        error_messages.append('Email cannot be empty.')
    elif not re.match(r"[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}$", email):
```

```
        error_messages.append('Please enter a valid email address.')

    if not country_code:
        error_messages.append('Please select a country code.')

    if error_messages:
        error_divs = [html.Div(message, style=error_style) for message in
error_messages]
        return error_divs, None
    else:
        confirmation_message = (
            f'Form Successfully Submitted: \n'
            f"Full Name: {first_name} {last_name}, \nAge:{age} years old,
\nGender:{gender}, "
            f"\nAddress:{address}, \nPhone: {country_code} {phone_number},
\nEmail: {email} \nThank You for Signing Up!!!"
        )
        return None, dcc.ConfirmDialog(
            displayed=True,
            message=confirmation_message,
        )


# ############################
# # PHASE 4 Callbacks
# ############################

#########################
# TABS Callback
#########################

@app.callback(
    Output(component_id='tabs-content', component_property='children'),
    [Input(component_id='tabs', component_property='value')]
)
def render_content(tab):
    if tab == 'Stays':
        return stays_layout
    elif tab == 'NormOut':
        return info_layout
    elif tab == 'Trend':
        return trend_analysis_layout
    elif tab == 'Stats':
        return stats_layout
    elif tab == 'Sign up':
        return sign_up_layout

# ############################
# # PHASE 5 Server
# ############################

app.server.run(
    debug = False,
    port = 8080,
    host = '0.0.0.0'
)
```

References

[1] Airbnb. (n.d.). The Emerging Markets Powering Airbnb's Global Growth Report. Airbnb Newsroom. https://news.airbnb.com/the-emerging-markets-powering-airbnbs-global-growth-report/

[2] Azmoudeh, A. (Year). Airbnb Open Data [Data set]. Kaggle. https://www.kaggle.com/datasets/arianazmoudeh/airbnbopendata/data

[3] The website "Statistics Easily" provides information on outlier detection and treatment. Retrieved from https://statisticseasily.com/outlier-detection-and-treatment/

[4] Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: A review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065), 20150202. https://doi.org/10.1098/rsta.2015.0202

[5] Wikipedia contributors. (2024, February 10). Normality test. Wikipedia. https://en.wikipedia.org/wiki/Normality_test

[6] Hudiburgh, L. M., & Garbinsky, D. (2020). Data visualization: Bringing data to life in an introductory statistics course. *Journal of Statistics Education*, 28(3), 262-279. https://doi.org/10.1080/10691898.2020.1796399

[7] Seaborn Development Team. (n.d.). Seaborn: Statistical data visualization. Retrieved from https://seaborn.pydata.org/tutorial/distributions.html

[8] Google LLC. (n.d.). Google Cloud Platform. Retrieved from https://console.cloud.google.com/

[9] Plotly Technologies Inc. (n.d.). Dash. Retrieved from https://dash.plotly.com/