Faculdade de Ciências da Universidade de Lisboa

Masters in Data Sciences

&

Masters in Informatics

Information Integration and Analytic Data Processing (2022/2023)

"Optimizing Airbnb's in Seattle"

Project Report

Group 6

Ana Rodrigues, FC30641

Ariana Dias, FC53687

Pedro Travessa, FC59479

Rita Rodrigues, FC54859

Index

List of Figures

1. List of Tables

2. Introduction

This project report was prepared in the context of the curricular unit "**Information Integration and Analytic Data Processing**". This project aims to explore ETL systems, dimensional modeling, and data analysis for decision-making in business-oriented processes.

Therefore, we positioned ourselves as a consulting company operating in **analytics-based operations** optimization and proposed to target the Airbnb business. **Airbnb** was founded in 2007 and has become a popular choice for travelers looking for an alternative to traditional hotels. The platform allows hosts to rent their properties from spare bedrooms to entire homes. With millions of listings worldwide, Airbnb is currently an influential force in the tourism and travel industry sectors.

In the context of this project, we selected datasets containing relevant information on Airbnb in Seattle (USA) and **Seattle** city. The goal is to find actionable insights that can generate value for potential customers, notably new or current Airbnb hosts.

3. Stage 1 – Data analysis and business definition

3.1. Data sources

This project draws on several datasets from Kaggle to comprehensively analyze Airbnb activity in Seattle.

The primary dataset used is the Seattle Airbnb Open Data, which provides detailed information about homestay listings in Seattle, WA. This dataset is part of the Airbnb Inside initiative and includes three sub-datasets: Listings, Reviews, and Calendar. Listings contain detailed information about each listing, including the host, the Airbnb, and average review scores. Reviews include reviews from June 7, 2009, to January 3, 2016, and each review is accompanied by a unique ID for the reviewer as well as their comments, while Calendar provides listing IDs and price and availability information for specific dates ranging from January 4, 2016, to January 2, 2017.

To explore the relationship between weather patterns and Airbnb bookings in Seattle, we utilized the dataset Did it rain in Seattle? (1948-2017). This dataset, compiled by NOAA, contains complete records of daily rainfall patterns and maximum and minimum temperatures at the Seattle-Tacoma International Airport from January 1, 1948, to December 12, 2017. The dataset is in the public domain.

We also used the Seattle Parks and Recreation Data to identify the locations of parks in Seattle and analyze their proximity to Airbnb listings. Specifically, we used the dataset 'seattle-parks-and-recreation-park-addresses.csv,' which includes the X and Y coordinates of Seattle Parks and Recreation Park addresses.

3.2. Data sources processing and preliminary ana

In this study, data analysis was conducted on five tables. The analysis process involved importing the data, exploring its characteristics, handling missing data, converting data types, and summarizing the data using descriptive statistics and plots.

3.2.1. Calendar

We started by importing the 'Calendar' dataset. **Figure 1** shows the head and tail of the DataFrame obtained from reading the 'the calendar.csv' file.

	listing_id	date	available	price
0	241032	2016-01-04	t	$85.00
1	241032	2016-01-05	t	$85.00
2	241032	2016-01-06	f	NaN
3	241032	2016-01-07	f	NaN
4	241032	2016-01-08	f	NaN
...	...	...	...	...
1393565	10208623	2016-12-29	f	NaN
1393566	10208623	2016-12-30	f	NaN
1393567	10208623	2016-12-31	f	NaN
1393568	10208623	2017-01-01	f	NaN
1393569	10208623	2017-01-02	f	NaN

1393570 rows × 4 columns

Figure 1 – Jupyter Notebook snapshot of the data frame (head) obtained from reading the 'calendar.csv' file

We proceeded to observe its information and discovered that the "price" column in the DataFrame contains missing values. Upon further investigation, we discovered that these missing values occurred only when the listing was marked as unavailable on the specified date. To handle these missing values, we decided to impute them with the string "Not available", which accurately reflects the reason for the missing values. We also checked for duplicates and found that there were no duplicates present in this DataFrame.

During the data type conversion step, we converted the "date" column in the DataFrame from a string to a datetime format. Additionally, we cleaned and converted the "price" column by removing the "$" sign and converting its values to float, except where the string "Not

available" was present. For the "available" column, we replaced the 't' values with 1 and the 'f' values with 0, to convert it from a Boolean to an integer format.

After performing the above steps, we exported the cleaned DataFrame. The next table summarizes data in the 'Calendar' table after cleaning and formatting.

Table 1 – Information contained in the 'Calendar' table after cleaning and formatting.

Field	Description	Data type	Example
listing_id	Id of the property	int64	241032
date	Date of the review provided by the customer	datetime64[ns]	04/01/2016
available	Availability status of the property on the specified date	int64	1
price	Daily price for property rental	object	85

We additionally computed some statistics to better understand the data.

Mean price for available listings	137.944859
Median price for available listings	109.0
Minimum price for available listings	10.0
Maximum price for available listings	1650.0
Number of available listings	934542
Total number of listings	1393570
Proportion of available listings	0.67061
Minimum date	2016-01-04 00:00:00
Maximum date	2017-01-02 00:00:00
Date range (days)	365

Figure 2 – Jupyter Notebook snapshot of descriptive statistics from the 'calendar' dataset.

To complement this analysis, we generated several plots to visualize the data and identify any trends or patterns.

Figure 3 – Jupyter Notebook snapshot of the plot on available Airbnb listings between 01-01-2026 and 01-01.2017

From the time series plot of available Airbnb listings over time, we can observe that the number of available listings is at its lowest at the beginning of the year, before rapidly increasing until April. It then stabilizes and drops again in June. After that, the number of available listings slowly grows until the end of the year. From the plot, we can infer that the number of available listings follows a seasonal trend, with higher availability in the spring and the summer months and towards the end of the year. Hosts possibly offer more listings during the warmer months and holiday season to take advantage of higher demand.

Figure 4 – Jupyter Notebook snapshot of the plot on the mean price of Airbnb listings between 01-01-2026 and 01-01.2017

Based on the plot, it can be observed that there is a clear seasonal trend in the mean prices of Airbnb listings. The trend shows that the prices are lowest at the beginning of the year, gradually increasing until they reach a peak in June. From June to September, the prices remain stable and consistent and then start to decrease again.

This pattern suggests that there may be external factors influencing the pricing trends of Airbnb listings. For instance, it could be due to the seasonal variation in demand for Airbnbs. In the winter months, there may be a lower demand, which could lead to lower prices. Conversely, in the summer months, demand for Airbnbs may be higher, leading to increased prices.

Figure 5 – Jupyter Notebook snapshot of the histogram of Airbnb listing prices between 01-01-2026 and 01-01.2017

The histogram of listing prices shows that the distribution is heavily skewed to the right, with a large concentration of listings at lower prices and a long tail of higher-priced listings. Specifically, most listings group at the lower end of the price range, with a peak in the first bin of the histogram, followed by a gradual decrease in the frequency of listings as the prices increase.

3.2.2. Listings

We proceeded with the analysis for the 'listings' dataset. **Figure 6** shows the head and tail of the DataFrame obtained from reading the 'listings.csv' file.

	id	listing_url	scrape_id	last_scraped	name	summary	space	description	experiences_offered	neig
0	241032	https://www.airbnb.com/rooms/241032	20160104002432	2016-01-04	Stylish Queen Anne Apartment	NaN	Make your self at home in this charming one-be...	Make your self at home in this charming one-be...	none	
1	953595	https://www.airbnb.com/rooms/953595	20160104002432	2016-01-04	Bright & Airy Queen Anne Apartment	Chemically sensitive? We've removed the irrita...	Beautiful, hypoallergenic apartment in an extr...	Chemically sensitive? We've removed the irrita...	none	wor
2	3308979	https://www.airbnb.com/rooms/3308979	20160104002432	2016-01-04	New Modern House-Amazing water view	New modern house built in 2013. Spectacular s...	Our house is modern, light and fresh with a wa...	New modern house built in 2013. Spectacular s...	none	U cl
3	7421966	https://www.airbnb.com/rooms/7421966	20160104002432	2016-01-04	Queen Anne Chateau	A charming apartment that sits atop Queen Anne...	NaN	A charming apartment that sits atop Queen Anne...	none	
4	278830	https://www.airbnb.com/rooms/278830	20160104002432	2016-01-04	Charming craftsman 3 bdm house	Cozy family craftman house in beautiful neighb...	Cozy family craftman house in beautiful neighb...	Cozy family craftman house in beautiful neighb...	none	
...	...	...	...	...	...	...	...	...	...	
3813	8101950	https://www.airbnb.com/rooms/8101950	20160104002432	2016-01-04	3BR Mountain View House in Seattle	Our 3BR/2BA house boasts incredible views of t...	Our 3BR/2BA house bright, stylish, and wheelch...	Our 3BR/2BA house boasts incredible views of t...	none	We
3814	8902327	https://www.airbnb.com/rooms/8902327	20160104002432	2016-01-04	Portage Bay View!-One Bedroom Apt	800 square foot 1 bedroom basement apartment w...	This space has a great view of Portage Bay wit...	800 square foot 1 bedroom basement apartment w...	none	qu
3815	10267360	https://www.airbnb.com/rooms/10267360	20160104002432	2016-01-04	Private apartment view of Lake WA	Very comfortable lower unit. Quiet, charming m...	NaN	Very comfortable lower unit. Quiet, charming m...	none	
3816	9604740	https://www.airbnb.com/rooms/9604740	20160104002432	2016-01-04	Amazing View with Modern Comfort!	Cozy studio condo in the heart on Madison Park...	Fully furnished unit to accommodate most needs...	Cozy studio condo in the heart on Madison Park...	none	
3817	10208623	https://www.airbnb.com/rooms/10208623	20160104002432	2016-01-04	Large Lakefront Apartment	All hardwood floors, fireplace, 65" TV with Xb...	NaN	All hardwood floors, fireplace, 65" TV with Xb...	none	

3818 rows × 92 columns

Figure 6 – Jupyter Notebook snapshot of the data frame (head and tail) obtained from reading the 'listings.csv' file; only the first columns are shown.

The dataset originally had 92 columns, but some of them were irrelevant to our analysis, such as the URLs or columns with identical location information for all listings. Therefore, we dropped them from the dataset. Additionally, we removed the 'security_deposit', 'cleaning_fee', and 'square_feet' columns, even though they were important for our analysis because they contained a high percentage of missing values (more than 20%). We also checked the DataFrame for duplicates and found none.

Regarding the modification of data types, we removed the percentage sign from the 'host_response_rate' and 'host_acceptance_rate' columns and converted them to float. We also converted the 'host_since', 'first_review', and 'last_review' columns to datetime format and cleaned the 'price' column using the same method as in the previous dataset. Additionally, we replaced the 't' values with 1 and 'f' values with 0 in the 'host_identity_verified', 'is_location_exact', and 'instant_bookable' columns. To handle missing data, we first analyzed the dataset and observed that some columns had a significant

percentage of missing values (between 13% and 20%, approximately). For these columns, we decided to impute the mean for numeric columns and the mode for the 'host_response_time' categorical column. For the 'first_review' and 'last_review' columns, which had 16.42% of missing values, we used the 'fill' method to impute missing values. For other columns with missing values, we concluded that these values would not significantly impact our analysis, and therefore eliminated all rows with missing values. After removing the missing values, we also converted the 'host_total_listing_count' and 'zipcode' columns to integer format. The 'zipcode' column contained errors in the form of line breaks, so we had to extract only the numerical values.

After completing these data-cleaning steps, we exported the cleaned DataFrame. The resulting dataset had a total of 3787 rows and 41 columns. **Table 2** summarizes data in the 'Listings' table after cleaning and formatting.

Table 2 – Information contained in the 'Listings' table after cleaning and formatting.

Field	Description	Data type	Example
id	Id of the property	int64	241032
name	Name of the property	string	Stylish Queen Anne Apartment
host_id	Id of the host	int64	956883
host_since	Date of host registry	datetime64[ns]	11/08/2011
host_response_time	The typical timeframe that the host takes to respond	float64	within a few hours
host_response_rate	Cumulative response rate (0-1) of the host	float64	0.96
host_acceptance_rate	Cumulative acceptance rate (0-1) of the host	float64	1
host_total_listings_count	Number of properties of a specific host	Int32	3
host_verifications	The process used by Airbnb to verify the identity of the property host	list	['email', 'phone', 'reviews', 'kba']
host_identity_verified	Has the host been verified? (Y/N)	float64	1
street	Street of the property	str	Gilman Dr W, Seattle, WA 98119, United States
neighbourhood_cleansed	specific neighborhood or area where the property is located	str	West Queen Anne
neighbourhood_group_cleansed	Larger area or region that contains multiple neighborhoods	str	Queen Anne
zipcode	Zip code of the property	int64	98119
latitude	Latitude of the property	float64	47.63628904
longitude	Longitude of the property	float64	-122.3710252
is_location_exact	Is the property location exact? (Y/N)	int64	1
property_type	Type of property	str	Apartment

Table 2 (cont.) – Information contained in the 'Listings' table after cleaning and formatting.

Field	Description	Data type	Example
room_type	Whether the property is just the room of the entire home/apartment	str	Entire home/apt
accommodates	Maximum number of guests	int64	4
bathrooms	Number of bathrooms on the property	float64	1
bedrooms	Number of bedrooms in the property	float64	1
beds	Number of beds in the property	float64	1
bed_type	Type of beds available on the property	str	Real Bed
amenities	Amenities available on the property	list	{TV,"Cable TV",Internet}
price	Daily property price	float64	85
neighbourhood_cleansed	Specific neighborhood or area where the property is located	str	West Queen Anne
neighbourhood_group_cleansed	Larger area or region of neighborhoods	str	Queen Anne
beds	Number of beds in the property	float64	1
bed_type	Type of beds available on the property	str	Real Bed
amenities	Amenities available on the property	list	{TV,"Cable TV",Internet}
price	Daily property price	float64	85
guests_included	How many guests are included in the advertised 'price'	int64	2
number_of_reviews	Number of reviews the property already has	int64	207
first_review	Date of the first review posted for the property	datetime64 [ns]	01/11/2011
last_review	Date of the last review posted for the property	datetime64 [ns]	02/01/2016
review_scores_rating	Review score on the overall appreciation	float64	95
review_scores_accuracy	Review score on how accurate the information about the property is	float64	10
review_scores_cleanliness	Review score on the cleanliness	float64	10
review_scores_checkin	Review score on the check-in process	float64	10
review_scores_communication	Review the score on the communication with the host	float64	10
review_scores_location	Review score on the property location	float64	9
review_scores_value	Review scores on the value of the booking/purchase	float64	10
instant_bookable	Is the property immediately bookable, i.e., no need to wait for the host to reply? (Y/N)	int64	0
cancellation_policy	The severitycancellationtion policy	str	moderate
reviews_per_month	The average number of reviews per month	float64	4.07

We calculated several statistics to gain insights into the Seattle Airbnb market. We focused on aspects of the listing, including the host, the neighborhood, the property, and the reviews. **Figures 7** and **Figure 8** show a snapshot of the calculated statistics.

A

	0
Number of Hosts	2740.000000
Average Host Response Rate	0.948916
Average Host Acceptance Rate	0.999670
Average Listings per Host	7.097703

B

	Frequency	Proportion
Broadway	391	0.103248
Belltown	227	0.059942
Wallingford	167	0.044098
Fremont	158	0.041722
Minor	135	0.035648
...	...	...
Arbor Heights	5	0.001320
Pinehurst	4	0.001056
South Beacon Hill	4	0.001056
South Park	3	0.000792
Roxhill	2	0.000528

C

	amin	amax	mean	median	std
accommodates	1.0	16.0	3.349617	3.0	1.975892
bathrooms	0.0	8.0	1.259440	1.0	0.590435
bedrooms	0.0	7.0	1.308424	1.0	0.883577
beds	1.0	15.0	1.737259	1.0	1.141075
price	20.0	1000.0	127.954581	100.0	90.374145

D

	property_type	room_type	bed_type
unique values	[Apartment, House, Cabin, Condominium, Camper/RV, Bungalow, Townhouse, Loft, Boat, Other, Dorm, Bed & Breakfast, Treehouse, Yurt, Chalet, Tent, nan]	[Entire home/apt, Private room, Shared room]	[Real Bed, Futon, Pull-out Sofa, Airbed, Couch]
mode	House	Entire home/apt	Real Bed

Figure 7 – Jupyter Notebook snapshots of the DataFrames (head and tail) obtained from the different statistics on Seattle's Airbnb properties market based on the date of: (a) the host, (b) the neighborhood, (c) the property rooms and price, and (d) the type of property.

	review_scores_rating	review_scores_accuracy	review_scores_cleanliness	review_scores_checki
count	3787.000000	3787.000000	3787.000000	3787.00000
mean	94.549809	9.636465	9.557548	9.78852
std	6.024127	0.636449	0.724772	0.53761
min	20.000000	2.000000	3.000000	2.00000
25%	94.000000	9.636392	9.000000	9.78670
50%	95.000000	10.000000	10.000000	10.00000
75%	98.000000	10.000000	10.000000	10.00000
max	100.000000	10.000000	10.000000	10.00000

	review_scores_communication	review_scores_location	review_scores_value	reviews_per_month
count	3787.000000	3787.000000	3787.000000	3787.000000
mean	9.811297	9.610398	9.453791	2.082490
std	0.509937	0.572361	0.682568	1.666338
min	2.000000	4.000000	2.000000	0.020000
25%	9.809599	9.000000	9.000000	0.840000
50%	10.000000	10.000000	9.452245	2.000000
75%	10.000000	10.000000	10.000000	2.665000
max	10.000000	10.000000	10.000000	12.150000

Figure 8 – Jupyter Notebook snapshots of the DataFrames (head and tail) obtained from the different statistics on Seattle's Airbnb properties market based on review scores data.

Figure 7A shows that there are 2,740 unique hosts, and on average, they have a high response rate of 94.89% and an acceptance rate close to 100%. On average, each host has 7.1 listings, indicating that many hosts have multiple properties listed on Airbnb.

Figure 7B shows the frequency and proportion of listings in different neighborhoods of Seattle. The first column lists the neighborhood names, while the second and third columns show the frequency and proportion of listings respectively. It appears that Broadway has the highest proportion of listings at 10.3%, followed by Belltown at 6.0%, and Wallingford at 4.4%. Meanwhile, some neighborhoods, such as Arbor Heights and Pinehurst, have relatively few listings, with a proportion of only 0.1% or less.

Figure 7C shows the summary statistics for the numerical columns of the property features. We can see that the mean price is $127.95, with a standard deviation of $90.37, and that the average number of accommodates is 3.35, with a standard deviation of 1.98.

Figure 7D displays the unique values and mode for the categorical columns of the property features. We can see that the most common property type is a house, the most common room type is an entire home/apartment, and the most common bed type is a real bed.

We further analyzed some selected visualizations of the listings dataset, namely the room type and scores distribution (**Figure 9**) and the geographical distribution of all properties (**Figure 10**).

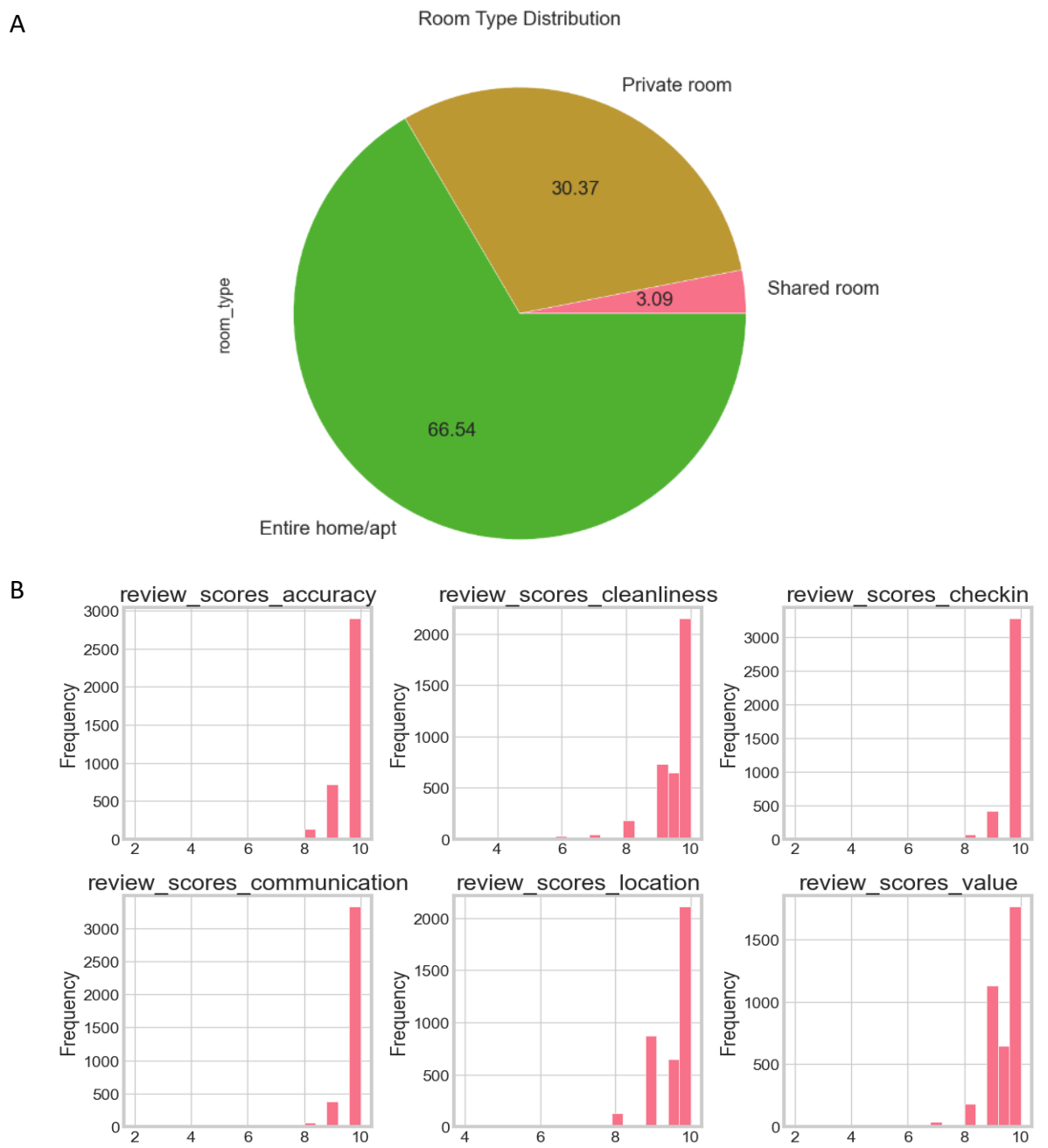


Figure 9 – Jupyter Notebook snapshots of (A) the pie chart representing the room type distribution and (B) the distribution of the scores in the 'listings' dataset

Figure 9A shows that the largest portion of the pie chart is taken up by "Entire home/apt" with 66.54%, which indicates that the majority of the listings in the dataset are for an entire home or apartment. "Private room" makes up the second largest portion of the chart with 30.37%, indicating that a significant portion of listings offer private rooms within a home or

apartment. The smallest portion of the chart is taken up by "Shared room" with only 3.09%, indicating that very few listings offer a shared room as an option.

Figure 9B shows that the majority of review scores' accuracy is above 8. This suggests that accuracy is not a significant issue for guests and is generally considered satisfactory. The cleanliness score is broader spreading, indicating a wider range of scores than accuracy. Most ratings are high, but smaller bars with lower ratings suggest some guests reported issues. The bars extend as low as 6, showing a wider range of opinions on cleanliness. Overall, guests are mostly satisfied, but some reported issues. The scores for check-in show a large bar at the high end, indicating that most guests have positive experiences. The smaller bars at the low end suggest a few negative experiences. Overall, check-in seems to be considered a positive aspect. For communication, the scores show a similar pattern to the histogram for check-in, with a significant bar located at the highest end of the histogram and smaller bars located at the lower end. This suggests that the majority of guests have reported positive experiences with communication during their stay, while very few guests have reported negative experiences. Regarding location, the plot has more bars than the others, indicating a wider range of scores. While most ratings are high, the presence of smaller bars suggests some guests have less favorable opinions. Finally, the overall review scores show high bars towards the highest end of the histogram, indicating that the majority of guests have reported positive experiences with the value of the properties. This suggests that guests feel that the properties offer good value for their money.

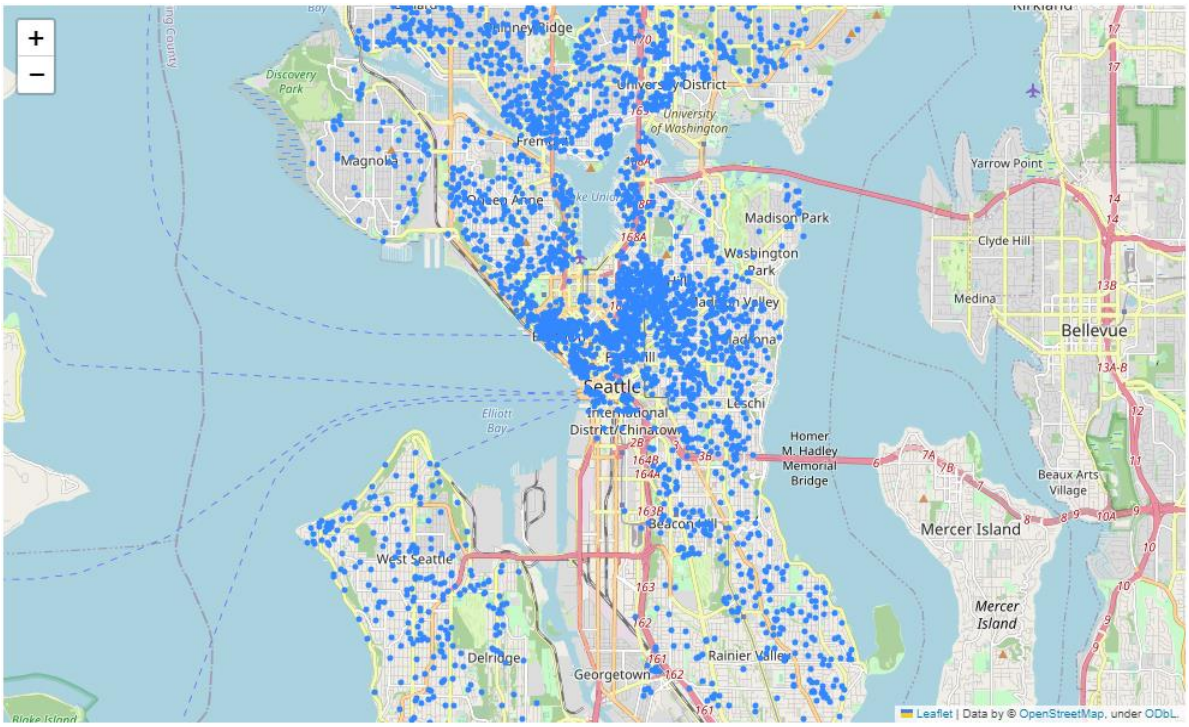


Figure 10 – Jupyter Notebook snapshots of geographical map distribution of the properties based on the 'listings' dataset.

Figure 10 shows a good distribution of the properties across the entire Seattle, with some concentration on downtown, Fremont, and the University District.

3.2.3. Reviews

Next, we imported and examined the 'reviews' dataset. **Figure 11** shows the head of the DataFrame obtained from reading the 'review.csv' file.

Next, we checked the dataset's information using the info() method. The missing data only existed in the comment's column, and it was just 18 rows out of 84849. Therefore, we dropped those rows to clean the data. We also checked that there were no duplicates in the data and converted the date column to datetime format. **Table 3** summarizes data in the 'Reviews' table after cleaning and formatting.

	listing_id	id	date	reviewer_id	reviewer_name	comments
0	7202016	38917982	2015-07-19	28943674	Bianca	Cute and cozy place. Perfect location to everything!
1	7202016	39087409	2015-07-20	32440555	Frank	Kelly has a great room in a very central location. \r\nBeautiful building , architecture and a style that we really like. \r\nWe felt guite at home here and wish we had spent more time.\r\nWent for a walk and found Seattle Center with a major food festival in progress. What a treat.\r\nVisited the Space Needle and the Chihuly Glass exhibit. Then Pikes Place Market. WOW. Thanks for a great stay.
2	7202016	39820030	2015-07-26	37722850	Ian	Very spacious apartment, and in a great neighborhood. This is the kind of apartment I wish I had!\r\n\r\nDidn't really get to meet Kelly until I was on my out, but she was always readily available by phone. \r\n\r\nI believe the only "issue" (if you want to call it that) was finding a place to park, but I sincerely doubt its easy to park anywhere in a residential area after 5 pm on a Friday
3	7202016	40813543	2015-08-02	33671805	George	Close to Seattle Center and all it has to offer - ballet, theater, museum, Space Needle, restaurants of all ilk just blocks away, and the Metropolitan (probably the coolest grocer you'll ever find). Easy to find and Kelly was warm, welcoming, and really interesting to talk to.
4	7202016	41986501	2015-08-10	34959538	Ming	Kelly was a great host and very accommodating in a great neighborhood. She has some great coffee and while I wasn't around much during my stay the time I spent interacting with her was very pleasant. \r\n\r\nThe apartment is in a great location and very close to the Seattle Center. The neighborhood itself has a lot of good food as well!

Figure 11 – Jupyter Notebook snapshot of the data frame (head only) obtained from reading the 'reviews.csv' file.

Table 3 – Information contained in the 'Reviews' table after cleaning and formatting.

Field	Description	Data type	Example
listing_id	Id of the properties listed on Airbnb in Seattle	int64	7202016
id	Id of the Airbnb customer (user) providing the review	int64	38917982
date	Date of the review provided by the customer	datetime64[ns]	19/07/2015
reviewer_id		int64	28943674
reviewer_name	Name of the customer (user)	str	Bianca
comments	Comments about the property/stay	str	Cute and cozy place. Perfect location to everything!

After cleaning the reviews dataset, we calculated some statistics. We counted the number of reviews for each reviewer (**Figure 12A**), analyzed the date range and the number of reviews over time (**Figure 12B**), and made a word cloud from the comments section.

We see that some reviewers had multiple stays (perhaps visitors coming often to Seattle for business) while others have just one stay (**Figure 12A**). Before mid-2014, the number of reviews was relatively low, with less than 100 reviews per day. However, around mid-2014, the number of reviews started to increase rapidly, reaching almost 600 reviews per day in the second third of 2015; after that, the number of reviews dropped again (**Figure 12B**). The word cloud (**Figure 12C**) represents the most frequent words in the comments of the reviews, meaning that the larger the word appears in the cloud, the more often it is mentioned in the reviews. We can see that the most used words in the comments of the reviews are "place", "stay", "Seattle", "apartment", "house", and "room". This suggests that guests are primarily commenting on the quality of the accommodation, such as the overall "place" and the specific type of accommodation, whether it be an "apartment", "house", or "room". Additionally, the fact that "Seattle" appears prominently in the word cloud may suggest that guests are also commenting on their experiences in the city itself, such as its attractions.

A

	reviewer_id	reviewer_name	num_reviews
1188	206203	Amanda	67
38154	15121499	Kathryn	32
11565	2734499	Jason	19
19977	5775807	David	19
58391	29590276	Gerd	18
...	...	...	...
26832	8574337	Lauren	1
26833	8575076	Christina	1
26834	8575094	Garett	1
26835	8575117	Cynthia	1
75716	52812740	Arin	1

B

C

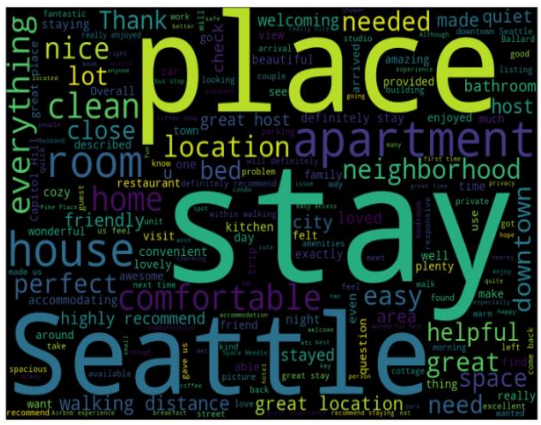


Figure 12 – Jupyter Notebook snapshots of (A) the DataFrame (head only) obtained from counting reviews per user, (B) plot on the number of reviews over time, and (C) word cloud from the 'comments' column.

3.2.4. Parks

We proceeded with importing the parks' dataset. **Figure 13A** shows the head of the DataFrame obtained from reading the 'review.csv' file. Subsequently, we performed a quick assessment of the data quality and found no missing values or duplicates. However, we noticed that the column "Location 1" contained redundant information from the "X Coord" and "Y Coord" columns, so we decided to remove it from the dataset. We also visualized the park locations on a map using the latitude and longitude coordinates provided in the dataset (**Figure 13B**). Finally, we exported the cleaned dataset. **Table 4** summarizes data in the 'Parks'' table after cleaning and formatting.

A

	PMAID	LocID	Name	Address	ZIP Code	X Coord	Y Coord	Location 1
0	281	2545	12th and Howe Play Park	1200 W Howe St	98119	-122.372985	47.636097	{'longitude': '-122.372985', 'needs_recoding':...
1	4159	2387	12th Ave S Viewpoint	2821 12TH Ave S	98144	-122.317765	47.577953	{'longitude': '-122.317765', 'needs_recoding':...
2	4467	2382	12th Ave Square Park	564 12th Ave	98122	-122.316455	47.607427	{'longitude': '-122.316455', 'needs_recoding':...
3	4010	2546	14th Ave NW Boat Ramp	4400 14th Ave NW	98107	-122.373536	47.660775	{'longitude': '-122.373536', 'needs_recoding':...
4	296	296	3001 E Madison	3001 E Madison St	98112	-122.293173	47.625169	{'longitude': '-122.293173', 'needs_recoding':...

B

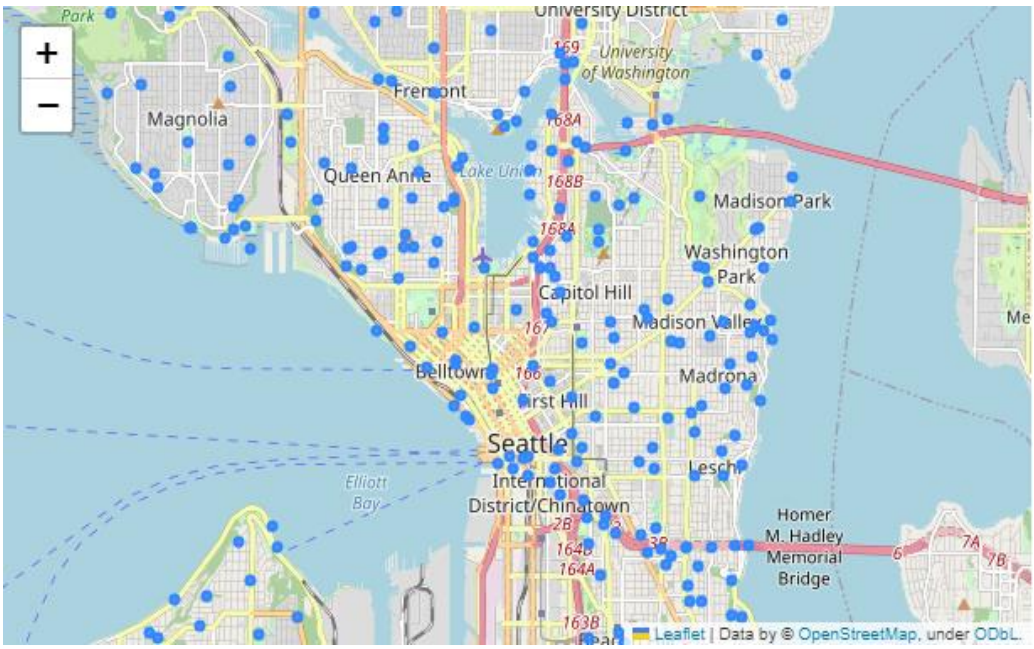


Figure 13 – Jupyter Notebook snapshot of (A) the data frame (head only) obtained from reading the 'parks.csv' file and (B) of geographical map distribution of Seattle parks in the 'parks' dataset.

Table 4 – Information contained in the 'Parks' table after cleaning and formatting.

Field	Description	Data type	Example
PMAID	Property management if	int64	281
LocID	Location id	int64	2545
Name	Park name	str	12th and Howe Play Park
Address	Park address	str	1200 W Howe St
ZIP Code	Zip code of park location	int64	98119
X Coord	Longitude of park location	float64	-122.373
Y Coord	Latitude of park location	float64	47.6361

3.2.5. Weather

We proceeded with importing and inspecting the weather data (**Figure 14A**). We found that there were only 3 rows with missing values, which we removed, and that there were no duplicates in the data. We converted the "DATE" column to datetime format and the "RAIN" column to an integer type, where 1 indicates that it rained and 0 indicates that it did not rain. Some descriptive statistics were also computed (**Figure 14B**). Finally, we exported the cleaned data. **Table 5** summarizes data in the 'Weather' table after cleaning and formatting.

A

	DATE	PRCP	TMAX	TMIN	RAIN
0	1948-01-01	0.47	51	42	True
1	1948-01-02	0.59	45	36	True
2	1948-01-03	0.42	45	35	True
3	1948-01-04	0.31	45	34	True
4	1948-01-05	0.17	45	32	True
...	...	...	...	...	...
25546	2017-12-10	0.00	49	34	False
25547	2017-12-11	0.00	49	29	False
25548	2017-12-12	0.00	46	32	False
25549	2017-12-13	0.00	48	34	False
25550	2017-12-14	0.00	50	36	False

B

	PRCP	TMAX	TMIN
count	25548.000000	25551.000000	25551.000000
mean	0.106222	59.544206	44.514226
std	0.239031	12.772984	8.892836
min	0.000000	4.000000	0.000000
25%	0.000000	50.000000	38.000000
50%	0.000000	58.000000	45.000000
75%	0.100000	69.000000	52.000000
max	5.020000	103.000000	71.000000

Total rainfall	2713.750000
Number of rainy days	10900.000000
Maximum daily rainfall	5.020000
Minimum daily rainfall	0.000000
Average daily rainfall	0.106222

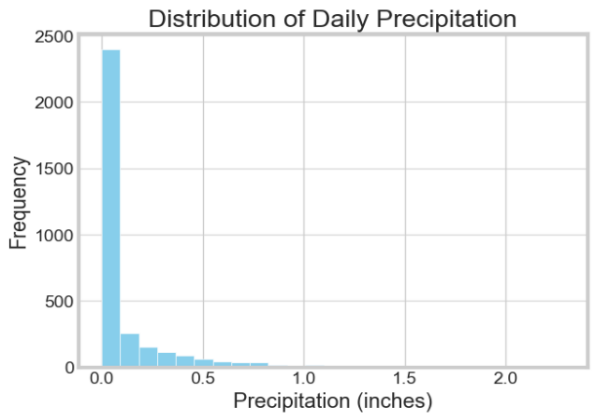
Figure 14 – Jupyter Notebook snapshot of (A) the data frame (head and tail) obtained from reading the 'weather.csv' file and (B) descriptive statistics on the weather dataset.

Table 5 – Information contained in the 'Weather' table after cleaning and formatting.

Field	Description	Data type	Example
DATE	Date	datetime64[ns]	01/01/1948
PRCP	Precipitation levels	float64	0.47
TMAX	Maximum temperature	int64	51
TMIN	Minimum temperature	int64	42
RAIN	Did it rain on that day? (Y/N)	int32	1

We further analyzed daily precipitation values since 2009 (**Figure 15A**) and average daily temperature between 2009 and 2017 (**Figure 15B**).

A



B

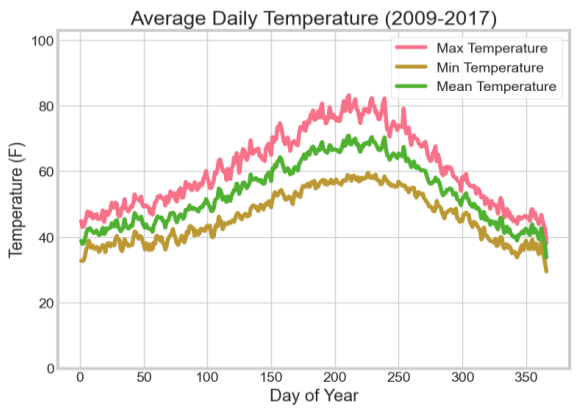


Figure 15 – Jupyter Notebook snapshot of (A) daily precipitation distribution and (B) average temperature based on the weather dataset.

The histogram (**Figure 15A**) shows that the distribution of daily precipitation values since 2009 is heavily skewed to the right, with a large number of days having very low precipitation amounts and a few days having much higher precipitation amounts. The plot of the average daily temperature between 2009 and 2017 (**Figure 15B**) shows three lines representing the maximum, minimum, and mean temperature. From day 150 until 250, the temperatures are higher, and the difference between the maximum and minimum temperatures is more pronounced. This suggests that this period experienced a higher level of temperature variability than other periods during the year. Additionally, the mean temperature line indicates that the temperatures during this time were consistently higher than average.

3.2.6. Connection between the different datasets

Overall, the information from the five tables used in this project can be connected as shown in **Figure 16**.

Figure 16 – Connection and links between the datasets used in this project.

3.3. Business Process

3.3.1. Positioning & business goals

Airbnb has revolutionized the way people travel and stay throughout the world with its innovative online platform. This platform connects hosts who want to rent out their homes, apartments, or rooms to travelers seeking an alternative to traditional hotel accommodations. One of the key benefits of using Airbnb is that it can be **less expensive** than hotels, with **lower rates** for rentals and the option to prepare meals rather than eat out. Additionally, travelers can enjoy unique accommodations with distinct personalities and **personalized experiences**. However, some travelers prefer the comfort and reliability of hotels, which typically offer uniform levels of service, facilities, and safety. Airbnb operates on a **commission-based business model**, charging hosts and guests a fee based on a percentage of each booking. The platform also provides various services such as expert photography, cleaning, and vacation experiences to enhance the booking process.

Our position is to operate like a company that provides consulting services to **Airbnb hosts in Seattle**, helping them enhance their listings and fully leverage the platform's potential. We understand that pricing is a crucial factor for hosts, which is why we offer guidance and support to optimize their listings for profitability. Through our **streamlined business mode**l and **data-driven approach**, we operate efficiently and cost-effectively, delivering maximum value to our clients.

Our services enable hosts to make **informed decisions** about their listings, pricing, and guest experience by analyzing **market trends** and **seasonal fluctuations**, as well as conducting **sentiment analysis** of the comments in the reviews dataset. Using advanced data analysis techniques, **we provide** valuable insights that help hosts fine-tune their pricing strategy, improve guest satisfaction, and gain a competitive advantage. Our **ultimate goal** is to help Airbnb hosts in Seattle increase their revenue, enhance their guest experience, and achieve long-term success on the platform.

3.3.2. Analytical questions

Given our business process and our operations, we will be able to answer a few analytical questions to translate into services. Based on these answers, hosts will be able to improve their and their guests' experience on the platform.

1. What factors are associated with higher/lower prices for Airbnb listings? For example, does having more bedrooms, better ratings, or a certain location increase the price of a listing?
2. How are the properties located in Seattle? Are there specific neighborhoods where Airbnbs are more common? How can these neighborhoods be described?
3. How does the weather influence the number of reservations? And the weekday?

4. Stage 2 – Dimensional modeling

Following the stage of data import and analysis and the business process conception, we proceeded to the stage of dimensional modeling. We first conceived the dimensions table and the fact table. In between, we identified and improved some aspects relative to the first stage. Namely, in the context, binary values were replaced by informative text. For example, in the 'available' column of the calendar table, we replaced 't' and 'f' with 'Available' and 'Not available'. The same strategy was employed for 'host_identity_verified', 'is_location_exact', and 'instant_bookable' of the listings table and for the column ‘Rain’ in the weather table.

4.1. Dimensions tables

Dimension tables are a fundamental part of a dimensional modeling approach in data warehousing. They store the descriptive information that provides context and meaning to the data in a fact table. Each dimension table represents a specific entity or aspect of the business, such as customers, products, time, or geographic locations. Dimension tables typically contain a primary key that uniquely identifies each record, along with attributes that provide additional details about the entity, such as name, description, category, or hierarchy. The data contained in these tables are used for data analysis, filtering, grouping, and joining with fact tables to answer business questions.

Overall, we defined 9 dimension tables as described next. When it is specified “from ‘tableName’”, it indicates that data is directly obtained from the respective table, whereas when it is specified “from ‘tableName’ processing”, it indicates that some data processing (transformation) will be required during the ETL process. Slowly changing dimension techniques to be employed are also indicated.

The following hierarchies were identified: i) in the ‘Date’ dimension, year ⇨ month ⇨ day, and ii) in the ‘Location’ dimension neighbourhoodGroupCleansed ⇨ neighbourhoodCleansed.

Then, the possibilities generated by different combinations of items in the measure ‘amenities’ would make the dimension ‘Facilities’ become a monster dimension. Therefore, we opted to implement an outrigger for this measure, turning it into a separate dimension linked to the ‘Facilities’ dimension.

Table 6 – Date dimension table

Attributes	Source	Observations
datePK	autoincremental	
date	User-defined	start at the first review and end at the last review
year	automatically derived from the date	1st in hierarchy
month	automatically derived from the date	2nd in hierarchy
day	automatically derived from the date	3rd in hierarchy
weatherSeason	automatically derived from the date	summer/spring/autumn/winter
weekDay	automatically derived from the date	weekday/not weekday
weekendIndicator	automatically derived from the date	weekend/not weekend
tourismSeason	from 'calendar' processing	high/low

Table 7 – Host dimension table

Attributes	Source	Observations
hostPK	autoincremental	-
hostID	from listings	-
hostSince	from listings	-
hostResponseTime	from listings	Apply SCD2 at the ETL stage
hostResponseRate	from listings	Apply SCD2 at the ETL stage
hostAcceptanceRate	from listings	Apply SCD2 at the ETL stage
hostTotalListings	from listings	Apply SCD2 at the ETL stage
hostVerifications	from listings	Apply SCD1 at ETL stage
hostIdentityVerified	from listings	Apply SCD1 at the ETL stage

Table 8 – Property dimension table

Attributes	Source	Observations
propertyPK	autoincremental	-
propertyID	from listings	-
propertyName	from listings	Apply SCD2 at the ETL stage
numberOfReviews	from listings	Apply SCD2 at the ETL stage
reviewScoresRating	from listings	Apply SCD2 at the ETL stage
reviewScoresAccuracy	from listings	Apply SCD2 at the ETL stage
reviewScoresCleanliness	from listings	Apply SCD2 at the ETL stage
reviewScoresCheckIn	from listings	Apply SCD2 at the ETL stage
reviewScoresCommunication	from listings	Apply SCD2 at the ETL stage
reviewScoresLocation	from listings	Apply SCD2 at the ETL stage
reviewScoresValue	from listings	Apply SCD2 at the ETL stage
reviewsPerMonth	from listings	Apply SCD2 at the ETL stage

Table 9 – Location dimension table

Attributes	Source	Observations
streetPK	autoincremental	-
street	from listings	-
neighbourhoodGroupCleansed	from listings	1st in hierarchy
neighbourhoodCleansed	from listings	2nd in hierarchy
isLocationExact	from listings	-
zipcode	from listings	-
latitude	from listings	-
longitude	from listings	-
nearbyParksCount	from 'parks' processing	-

Table 10 – Facilities dimension table

Attributes	Source	Observations
facilitiesPK	autoincremental	-
propertyType	from listings	-
roomType	from listings	-
accommodates	from listings	-
bathrooms	from listings	-
bedrooms	from listings	-
beds	from listings	-
bedType	from listings	-
guestsIncluded	from listings	-
amenities	PK from amenities outrigger	Outrigger to 'amenities' dimension

Table 11 – Price & booking dimension table.

Attributes	Source	Observations
nigthPricePK	autoincremental	
nightPriceRange	from listings	Make ranges at the ETL stage: 0-49.99; 50-99.99, 100-149.99, 150-199.99, 200-299.99, >300 Apply SCD2 at the ETL stage
instantBookable	from listings	Apply SCD1 at the ETL stage
cancelation Policy	from listings	-

Table 12 – Customer dimension table

Attributes	Source	Observations
reviewerPK	autoincremental	-
reviewerID	from reviews	-
reviewerName	from reviews	-

Table 13 – Amenities outrigger dimension table

Attributes	Source	Observations
amenitiesPK	autoincremental	-
24-Hour Check-in	from listings	Apply SCD1 at the ETL stage
Air Conditioning	from listings	Apply SCD1 at the ETL stage
Breakfast	from listings	Apply SCD1 at the ETL stage
Buzzer/Wireless Intercom	from listings	Apply SCD1 at the ETL stage
Cable TV	from listings	Apply SCD1 at the ETL stage
Carbon Monoxide Detector	from listings	Apply SCD1 at the ETL stage
Cat(s)	from listings	Apply SCD1 at the ETL stage
Dog(s)	from listings	Apply SCD1 at the ETL stage
Doorman	from listings	Apply SCD1 at the ETL stage
Dryer	from listings	Apply SCD1 at the ETL stage
Elevator in Building	from listings	Apply SCD1 at the ETL stage
Essentials	from listings	Apply SCD1 at the ETL stage
Family/Kid Friendly	from listings	Apply SCD1 at the ETL stage
Fire Extinguisher	from listings	Apply SCD1 at the ETL stage
First Aid Kit	from listings	Apply SCD1 at the ETL stage
Free Parking on Premises	from listings	Apply SCD1 at the ETL stage
Gym	from listings	Apply SCD1 at the ETL stage
Hair Dryer	from listings	Apply SCD1 at the ETL stage
Hangers	from listings	Apply SCD1 at the ETL stage
Heating	from listings	Apply SCD1 at the ETL stage
Hot Tub	from listings	Apply SCD1 at the ETL stage
Indoor Fireplace	from listings	Apply SCD1 at the ETL stage
Internet	from listings	Apply SCD1 at the ETL stage
Iron	from listings	Apply SCD1 at the ETL stage
Kitchen	from listings	Apply SCD1 at the ETL stage
Laptop Friendly Workspace	from listings	Apply SCD1 at the ETL stage
Lock on Bedroom Door	from listings	Apply SCD1 at the ETL stage
Other pet(s)	from listings	Apply SCD1 at the ETL stage
Pets Allowed	from listings	Apply SCD1 at the ETL stage
Pets live on this property	from listings	Apply SCD1 at the ETL stage
Pool	from listings	Apply SCD1 at the ETL stage
Safety Card	from listings	Apply SCD1 at the ETL stage
Shampoo	from listings	Apply SCD1 at the ETL stage
Smoke Detector	from listings	Apply SCD1 at the ETL stage
Smoking Allowed	from listings	Apply SCD1 at the ETL stage
Suitable for Events	from listings	Apply SCD1 at the ETL stage
TV	from listings	Apply SCD1 at the ETL stage
Washer	from listings	Apply SCD1 at the ETL stage
Washer / Dryer	from listings	Apply SCD1 at the ETL stage
Wheelchair Accessible	from listings	Apply SCD1 at the ETL stage
Wireless Internet	from listings	Apply SCD1 at the ETL stage

Table 14 – Weather dimension table

Attributes	Source	Observations
weatherPK	autoincremental	-
tMinBand	from wheather processing	Make ranges at ETL stage: >-10ºC, -9-0ºC, 1-10ºC, 11-20ºC, 21-30ºC, >=31ºC
tMaxBand	from wheather processing	Make ranges at ETL stage: >-10ºC, -9-0ºC, 1-10ºC, 11-20ºC, 21-30ºC, >=31ºC
precipitationBand	from wheather processing	Make ranges at the ETL stage: 0-0.99, 1-2.99, 3-4.99, >=5
rain	from weather	-

4.2. Fact table

A fact table is a central table in a data warehouse that stores quantitative data about business events, such as sales or customer counts. It connects to dimension tables through foreign keys, which reference the primary keys of the dimension tables. Each fact in the fact table is identified by the value of the primary key of the fact table. The granularity of a fact table determines the level of detail of the data stored in the table. The finer the granularity, the more dimensions and attributes the fact table has.

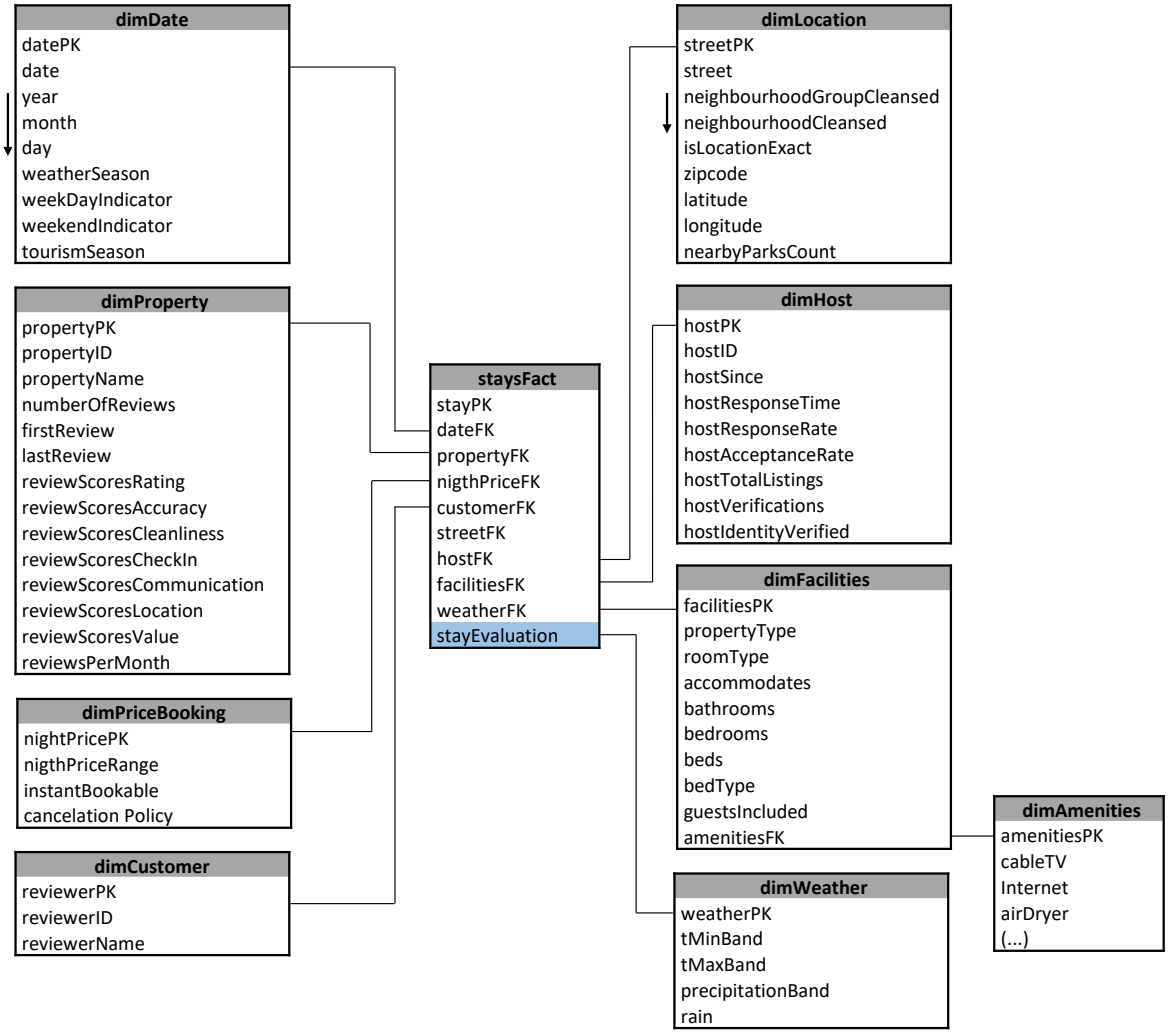
The fact table in our business is of 'transactional type' where the transaction is the stay. Therefore, our business is modeled at the **grain level of one stay, enjoyed by one customer, in one property, on a specific date**.

The fact table contains the foreign keys *stayFK, dateFK, propertyFK, nightPriceFK, customerFK streetFK, hostFK, facilitiesFK, weatherFK,* and the following metric *stayEvaluation*, which reflects an overall rating of the property and stay experience based on sentiment analysis.

It is noteworthy that one common metric that could be expected in our fact table and that would, indeed, be useful in our business process would be the total stay value. However, we are not able to calculate this metric since there is no indication in the dataset of the number of nights each customer stayed. Therefore, the night price was kept as an attribute in the dimension 'Price & booking'.

4.3. Star schema

This dimensional model from the dimensions and fact tables described above resulted in the star schema with one outrigger represented in **Figure 17**:



Notes

Arrows highlight top to down hierarchy in the associated measures
Blue highlights numerical measurements in the fact table

Figure 17 – Star schema resulting from the dimensional modeling for our business.

5. Stage 3 – ETL system and reports

In the world of data analysis and business intelligence, an ETL (Extract, Transform, Load) system is responsible for collecting, organizing, and preparing data for reporting and analysis. Having completed the dimensional modeling phase, we are now ready to utilize the ETL system to extract data from various sources, transform it into a usable format, and load it into our target database or data warehouse.

The ETL system acts as a bridge between different data sources such as operational systems, databases, and external APIs. It enables comprehensive data gathering and integration from these different sources. The data goes through transformations, including cleansing, validation, and applying business rules, to ensure accuracy and consistency. Finally, the transformed data is loaded into a database or data warehouse, where it forms the basis for generating reports and conducting analyses.

Figure 18 shows an overview of the ETL process implemented in this project. In the following sections, we will explore the key components and functionalities of our ETL system, highlighting its role in supporting our reporting and analytical requirements.

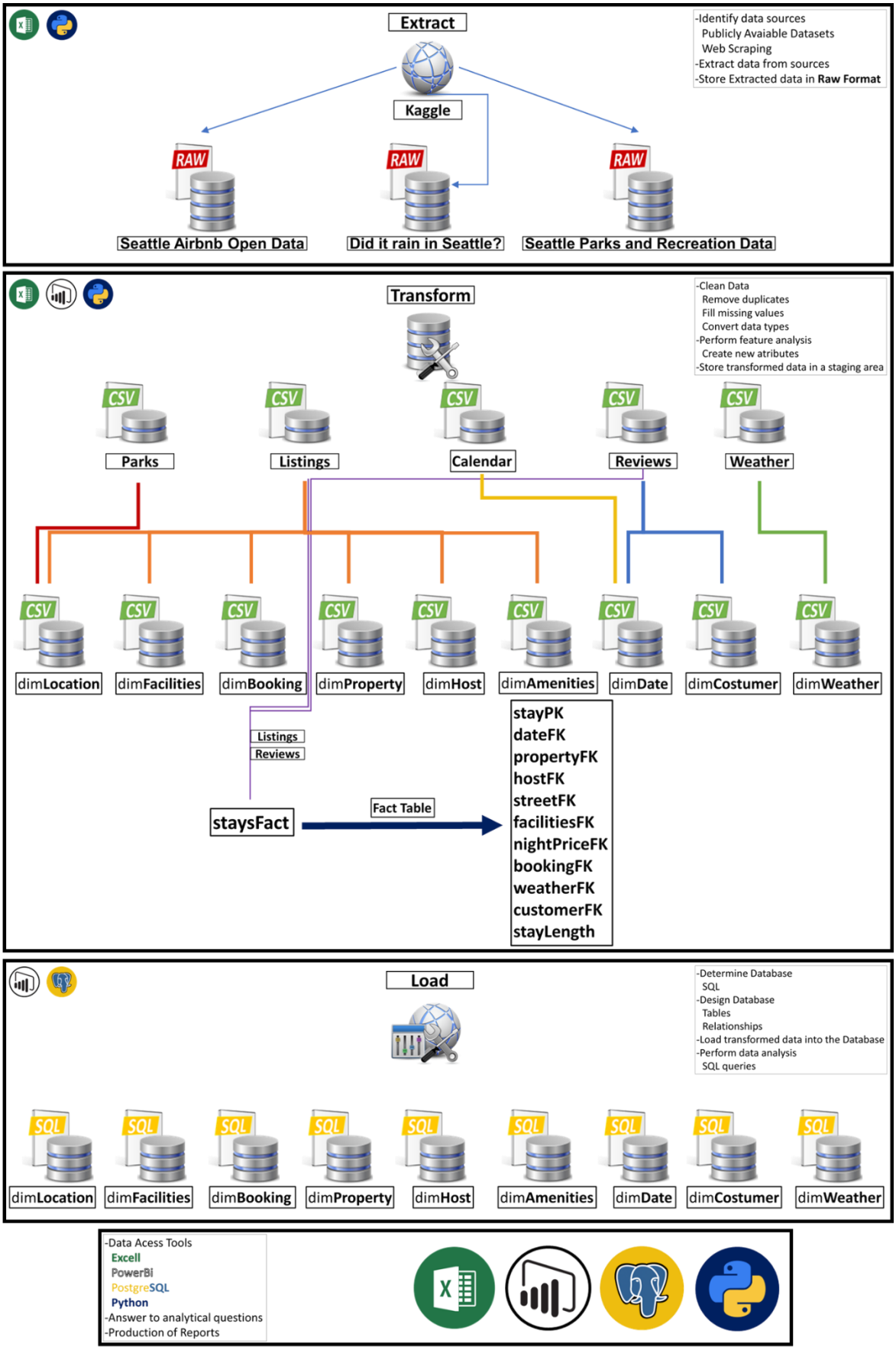


Figure 18 – Overview of the ETL process implemented in this project.

5.1. Extraction

In the extraction phase, we accessed a set of chosen datasets from Kaggle, a well-known platform for acquiring and sharing datasets. The data was distributed across five CSV files, conveniently stored within the "data" folder. By extracting data from these diverse sources, we aim to obtain a comprehensive understanding of the Seattle Airbnb market, taking into account factors such as weather conditions and the availability of nearby parks to enrich our analysis

5.2. Transformation

In the transformation phase of the ETL system, the data undergoes a series of operations within the data staging area (cleaned_data folder) to standardize and conform it, such as cleansing data errors, merging duplicates, and handling exceptions to ensure a uniform and consistent dataset. In our case, the data from the five files was manipulated in Python environment. These aspects of the transformation step have been completed in the first stage of this project (Data Analysis section of this report). The specific Notebook used for transformation has the name "IPAIG6Stage1.ipynb", and the resulting transformed data is stored in the "cleaned_data" folder.

Additional aspects of the transformation step executed at this third stage were the construction of dimension and fact tables. These transformations were guided by the star schema defined in the second stage of this project (**Figure 17**). The dimension tables capture descriptive attributes that provide context for analysis. These attributes may include time, location, property details, weather conditions, and other relevant dimensions specific to the Airbnb dataset. The fact table is constructed to store quantitative measures and numerical data related to Airbnb bookings. This includes information such as stay length and stay evaluation. The fact table serves as the central repository for these metrics, enabling aggregations and correlations between dimensions to derive meaningful insights.

The Notebook entitled "IPAIG6Stage3_ETL" contains the processes involved in constructing the dimension tables and the fact table. This Notebook takes the cleaned data stored in the "cleaned_data" folder as input and generates the respective tables as CSV files in the "dimensional_modeling" folder. Each process within the notebook focuses on building a specific table, and in the following sections, we will provide a detailed explanation of the steps involved for each of these tables.

5.2.1. Implementing slowly changing dimensions and indexes

Before starting with the ETL, we made a critical analysis of the most appropriate techniques to implement SCDs and indexes. Indexes can reveal useful for improving queries. However, they come at a cost of space in disk and a more cumbersome data loading into the data warehouse. Therefore, some of these indexes, namely clustered indexes and B+ trees, may only reveal truly useful in the long run, when the accumulation of data is such that querying

time justifies the extra storage space. In the context of this project, we only showcase the implementation of the B+ tree index for the attribute month in the Date dimension the and hash-based index for the Amenities outrigger dimension. Since PostgreSQL does not support bitmap indexing, we could not showcase the implementation of this type of index.

Appropriate indexes for the different attributes are described in **Tables 15** to **23**. Appropriate herein refers to the best indexes considering the corresponding data type and nature, which does not necessarily mean that such an index will be useful. For example, clustered indexes are appropriate for every primary key, which does not mean that they are useful and should be implemented in all dimensions. Another example: latitude and longitude data are indexable by B+ trees but it is unlikely that common queries will use latitude and longitude filters to justify the implementation of such index for these attributes.

Table 15 – SDC and indexes in the Date dimension

Attributes	Appropriate SDC	Appropriate index
datePK	None	clustered
date	None	clustered
year	None	B+
month	None	B+
day	None	B+
weatherSeason	None	bitmap
weekDay	None	bitmap
weekendIndicator	None	bitmap
tourismSeason	None	bitmap

Table 16 – SDC and indexes in Host dimension

Attributes	Appropriate SCD	Appropriate index
hostPK	None	clustered
hostID	None	clustered
hostSince	None	B+ tree
hostResponseTime	SCD2	B+ tree
hostResponseRate	SCD2	B+ tree
hostAcceptanceRate	SCD2	B+ tree
hostTotalListings	SCD2	B+ tree
hostVerifications	SCD1	bitmap
hostIdentityVerified	SCD1	bitmap

Table 17 – SDC and indexes in Property dimension

Attributes	Appropriate SCD	Appropriate index
propertyPK	-	clustered
propertyID	-	clustered
propertyName	SCD2	-
numberOfReviews	SCD2	B+ tree
reviewScoresRating	SCD2	B+ tree
reviewScoresAccuracy	SCD2	B+ tree
reviewScoresCleanliness	SCD2	B+ tree
reviewScoresCheckIn	SCD2	B+ tree
reviewScoresCommunication	SCD2	B+ tree
reviewScoresLocation	SCD2	B+ tree
reviewScoresValue	SCD2	B+ tree
reviewsPerMonth	SCD2	B+ tree

Table 18 – SDC and indexes in Location dimension

Attributes	Appropriate SCD	Appropriate index
streetPK	-	none
street	-	none
neighbourhoodGroupCleansed	-	none
neighbourhoodCleansed	-	none
isLocationExact	SCD1	bitmap
zipcode	-	B+ tree
latitude	-	B+ tree
longitude	-	B+ tree
nearbyParksCount	-	B+ tree

Table 19 – SDC and indexes in Facilities dimension

Attributes	Appropriate SCD	Appropriate index
facilitiesPK	-	clustered
propertyType	-	bitmap
roomType	-	bitmap
accommodates	-	B+ tree
bathrooms	-	bitmap
bedrooms	-	bitmap
beds	-	bitmap
bedType	-	bitmap
guestsIncluded	SCD1	B+ tree
amenities	-	-

Table 20 – SDC and indexes in Amenities outrigger dimension

Attributes	Source	Observations
amenitiesPK	-	clustered
24-Hour Check-in	SCD1	hashed
Air Conditioning	SCD1	hashed
Breakfast	SCD1	hashed
Buzzer/Wireless Intercom	SCD1	hashed
Cable TV	SCD1	hashed
Carbon Monoxide Detector	SCD1	hashed
Cat(s)	SCD1	hashed
Dog(s)	SCD1	hashed
Doorman	SCD1	hashed
Dryer	SCD1	hashed
Elevator in Building	SCD1	hashed
Essentials	SCD1	hashed
Family/Kid Friendly	SCD1	hashed
Fire Extinguisher	SCD1	hashed
First Aid Kit	SCD1	hashed
Free Parking on Premises	SCD1	hashed
Gym	SCD1	hashed
Hair Dryer	SCD1	hashed
Hangers	SCD1	hashed
Heating	SCD1	hashed
Hot Tub	SCD1	hashed
Indoor Fireplace	SCD1	hashed
Internet	SCD1	hashed
Iron	SCD1	hashed
Kitchen	SCD1	hashed
Laptop Friendly Workspace	SCD1	hashed
Lock on Bedroom Door	SCD1	hashed
Other pet(s)	SCD1	hashed
Pets Allowed	SCD1	hashed
Pets live on this property	SCD1	hashed
Pool	SCD1	hashed
Safety Card	SCD1	hashed
Shampoo	SCD1	hashed
Smoke Detector	SCD1	hashed
Smoking Allowed	SCD1	hashed
Suitable for Events	SCD1	hashed
TV	SCD1	hashed
Washer	SCD1	hashed
Washer / Dryer	SCD1	hashed
Wheelchair Accessible	SCD1	hashed
Wireless Internet	SCD1	hashed

Table 21 – SDC and indexes in Price & booking dimension

Attributes	Appropriate SCD	Appropriate index
nigthPricePK	-	clustered
nightPriceRange	SCD2	B+ tree
instantBookable	SCD1	bitmap
cancelation Policy	SCD2	bitmap

Table 22 – SDC and indexes in Weather dimension

Attributes	Appropriate SCD	Appropriate index
weatherPK	-	clustered
tMinBand	-	B+ tree
tMaxBand	-	B+ tree
precipitationBand	-	B+ tree
rain	-	bitmap

Table 23 – SDC and indexes in the Customer dimension

Attributes	Appropriate SCD	Appropriate index
reviewerPK	-	clustered
reviewerID	-	clustered
reviewerName	-	-

5.2.2. Process: the creation of Date dimension (dimDate)

Input sources of Date dimension: reviews.csv

The output of Date dimension: dimDate.csv

The code is used to construct the "dimDate" dimension table. It creates the "dimDate" DataFrame, which serves as the dimension table for date-related information.

The "date" column captures the complete date information, which is primarily useful in the data staging area. The code extracts unique values for year, month, and day from the "date" column. It further completes this dimension table by adding additional columns and defined in the dimensional modeling phase (**Figure 17**): i) the "weatherSeason" column categorizes the month into weather seasons such as Spring, Summer, Autumn, and Winter, i) the "weekDay" column represents the day of the week, with Monday assigned as 1 and Sunday as 7, iii) the "weekendIndicator" column indicates whether the date falls on a weekday or a weekend. This information is obtained based on the date and specific constraints defined in the code. For example, the code also determines if the date falls within the tourism season, which is defined as high between April to August and for one week before and after Christmas. based on information obtained from the calendar table in the first stage of the project (**Figure**

3). This information is stored in the "tourismSeason" column. To ensure proper organization, the DataFrame is sorted in ascending order based on the "date" column. An auto-incremental column ("datePK") is added as the primary key for the "dimDate" table. This is also what allows for correspondence between dates and the data stored in the "reviews" table.

Figure 19 shows the DataFrame that originates dimDate.csv.

	datePK	date	year	month	day	weatherSeason	weekDay	weekendIndicator	tourismSeason
0	1	2009-06-07	2009	6	7	Summer	7	weekend	high
1	2	2009-06-28	2009	6	28	Summer	7	weekend	high
2	3	2009-07-17	2009	7	17	Summer	5	weekday	high
3	4	2009-08-31	2009	8	31	Summer	1	weekday	high
4	5	2009-09-10	2009	9	10	Autumn	4	weekday	low
...	...	...	...	...	...	...	...	...	...
1924	1925	2015-12-30	2015	12	30	Winter	3	weekday	high
1925	1926	2015-12-31	2015	12	31	Winter	4	weekday	high
1926	1927	2016-01-01	2016	1	1	Winter	5	weekday	low
1927	1928	2016-01-02	2016	1	2	Winter	6	weekend	low
1928	1929	2016-01-03	2016	1	3	Winter	7	weekend	low

Figure 19 – Jupyter Notebook snapshot of the DataFrame (head and tail) containing the dimension Date. This DataFrame originates dimDate.csv.

5.2.3. Process: the creation of Property dimension (dimProperty)

Input sources of Property dimension: listings.csv

The output of Property dimension: dimProperty.csv

The code begins by selecting specific columns from the "listings" DataFrame that contain relevant information about the properties. These columns include "id," "name," "number_of_reviews," "first_review," "last_review," "review_scores_rating," "review_scores_accuracy," "review_scores_cleanliness," "review_scores_checkin," "review_scores_communication," "review_scores_location," "review_scores_value," and "reviews_per_month." To enhance the clarity and consistency of the dimension table, the code renames the selected columns in the "dimProperty" DataFrame using a dictionary-based mapping. For instance, "id" is renamed to "propertyID," "name" to "propertyName," and so on. **D**uplicate rows are then removed from the "dimProperty" DataFrame based on all columns, ensuring that each property is represented uniquely in the dimension table.

Next, the DataFrame is sorted in ascending order based on the "propertyID" column to maintain consistent ordering. An auto-incremental column called "propertyPK" is added to

the DataFrame, which assigns a unique identifier to each property and serves as the primary key for the "dimProperty" table.

Figure 20 shows the DataFrame that originates dimProperty.csv.

	propertyPK	propertyID	propertyName	numberOfReviews	firstReview	lastReview	reviewScoresRating	reviewScoresAccuracy
0	1	4291	Sunrise in Seattle Master Suite	35	2013-07-01	2015-10-18	92.000000	10.000000
1	2	5682	Cozy Studio, min. to downtown -WiFi	297	2010-03-21	2015-12-14	96.000000	10.000000
2	3	6606	Fab, private seattle urban cottage!	52	2009-07-17	2015-12-28	93.000000	9.000000
3	4	7369	launchingpad/landingpad	40	2009-06-07	2012-03-04	94.000000	10.000000
4	5	9419	Golden Sun vintage warm/sunny	79	2010-07-30	2015-12-07	91.000000	9.000000
...	...	...	...	...	...	...	...	...
3781	3782	10332096	Room & bath in suburban N Seattle	0	2014-08-13	2015-09-28	94.539262	9.636392
3782	3783	10334184	Historic Capitol Hill Garden Apt.	0	2014-09-06	2015-09-07	94.539262	9.636392
3783	3784	10339144	Studio in the heart of Capitol Hill	0	2012-08-05	2015-12-08	94.539262	9.636392
3784	3785	10339145	West Seattle Beachfront Apartment	0	2015-10-11	2015-11-29	94.539262	9.636392
3785	3786	10340165	Comfortable Reststop in Greenwood!	0	2015-09-26	2015-11-16	94.539262	9.636392

Figure 20 – Jupyter Notebook snapshot of the DataFrame (head and tail) containing the dimension Property; only the first columns are shown. This DataFrame originates dimProperty.csv.

5.2.4. Process: the creation of Host dimension (dimHost)

Input sources of Host dimension: listings.csv

The output of Host dimension: dimHost.csv

The code starts by selecting specific columns from the "listings" DataFrame that contain relevant information about the hosts. These columns include "host_id," "host_since," "host_response_time," "host_response_rate," "host_acceptance_rate," "host_total_listings_count," "host_verifications," and "host_identity_verified." To enhance clarity and consistency, the code renames the selected columns in the "dimHost" DataFrame using a dictionary-based mapping. For example, "host_id" is renamed to "hostID," "host_since" to "hostSince," and so on. Duplicate rows are then removed from the "dimHost" DataFrame based on all columns, ensuring that each host is represented uniquely in the dimension table.

Next, the code converts the "hostVerifications" column from a string representation of a list to the number of elements in each list. This provides a count of how many verifications each host has undergone. Next, the DataFrame is sorted in ascending order based on the "hostID" column to maintain consistent ordering. An auto-incremental column called "hostPK" is

added to the DataFrame, which assigns a unique identifier to each host and serves as the primary key for the "dimHost" table.

Figure 21 shows the DataFrame that originates dimHost.csv.

	hostPK	hostID	hostSince	hostResponseTime	hostResponseRate	hostAcceptanceRate	hostTotalListings	hostVerifications	hostIdentityVerified
0	1	4193	2008-11-10	within a few hours	0.880000	1.000000	4	5	Host Identity Verified
1	2	6207	2009-01-08	within an hour	1.000000	1.000000	1	6	Host Identity Verified
2	3	8021	2009-02-16	within a few hours	0.750000	1.000000	4	6	Host Identity Verified
3	4	8993	2009-03-03	within an hour	1.000000	1.000000	1	4	Host Identity Verified
4	5	11775	2009-03-30	within a few hours	1.000000	1.000000	1	5	Host Identity Verified
...	...	...	...	...	...	...	...	...	...
2735	2736	52990042	2016-01-01	within an hour	0.948868	0.999672	1	5	Host Identity Verified
2736	2737	53050379	2016-01-02	within an hour	0.948868	0.999672	1	3	Host Identity Not Verified
2737	2738	53065829	2016-01-02	within an hour	0.948868	0.999672	1	2	Host Identity Not Verified
2738	2739	53169216	2016-01-03	within an hour	0.948868	0.999672	1	2	Host Identity Not Verified
2739	2740	53208610	2016-01-03	within an hour	0.948868	0.999672	2	2	Host Identity Not Verified

Figure 21 – Jupyter Notebook snapshot of the DataFrame (head and tail) containing the dimension Host. This DataFrame originates dimHost.csv.

5.2.5. Process: the creation of Location dimension (dimLocation)

Input sources of Location dimension: listings.csv and parks.csv???

The output of Location dimension: dimLocation.csv

The code begins by selecting specific columns from the "listings" DataFrame that contain location-related information, such as "street," "neighbourhood_group_cleansed," "neighbourhood_cleansed," "is_location_exact," "zipcode," "latitude," and "longitude." Next, the code renames the columns in the "dimLocation" DataFrame to ensure clarity and consistency. Duplicate rows are then removed from the "dimLocation" DataFrame, ensuring each location is represented uniquely in the dimension table.

An auto-incremental column called "streetPK" is added to the DataFrame, which assigns a unique identifier to each street and serves as the primary key for the "dimLocation" table.

The code proceeds to define a radius (in kilometers) and a function named "calculate_distance." This function utilizes the Haversine formula to calculate the distance between two coordinates. The code iterates over each location in the "dimLocation" DataFrame and counts the number of nearby parks. It initializes the "nearbyParksCount" column to zero for each location. Within the iteration, the code compares the distance between each location's latitude and longitude coordinates with the latitude and longitude coordinates of parks. If the distance is within the defined radius, the park is considered nearby, and the count is incremented. Finally, the code updates the "nearbyParksCount" column in the "dimLocation" DataFrame with the calculated counts for each location.

Figure 22 shows the DataFrame that originates dimLocation.csv.

	streetPK	street	neighbourhoodGroupCleansed	neighbourhoodCleansed	isLocationExact	zipcode	latitude	longitude	nearbyParksCount
0	1	Gilman Dr W, Seattle, WA 98119, United States	Queen Anne	West Queen Anne	Location is Exact	98119	47.636289	-122.371025	8
1	2	7th Avenue West, Seattle, WA 98119, United States	Queen Anne	West Queen Anne	Location is Exact	98119	47.639123	-122.365666	8
2	3	West Lee Street, Seattle, WA 98119, United States	Queen Anne	West Queen Anne	Location is Exact	98119	47.629724	-122.369483	9
3	4	8th Avenue West, Seattle, WA 98119, United States	Queen Anne	West Queen Anne	Location is Exact	98119	47.638473	-122.369279	8
4	5	14th Ave W, Seattle, WA 98119, United States	Queen Anne	West Queen Anne	Location is Exact	98119	47.632918	-122.372471	9
...	...	...	...	...	...	...	...	...	...
3781	3782	Northwest 48th Street, Seattle, WA 98107, Unit...	Other neighborhoods	Fremont	Location is Exact	98107	47.664295	-122.359170	9
3782	3783	Fuhrman Avenue East, Seattle, WA 98102, United...	Capitol Hill	Portage Bay	Location is Exact	98102	47.649552	-122.318309	11
3783	3784	South Laurel Street, Seattle, WA 98178, United...	Rainier Valley	Rainier Beach	Location is Approximate	98178	47.508453	-122.240607	3
3784	3785	43rd Avenue East, Seattle, WA 98112, United St...	Capitol Hill	Madison Park	Location is Approximate	98112	47.632335	-122.275530	5
3785	3786	Westlake Avenue North, Seattle, WA 98109, Unit...	Queen Anne	East Queen Anne	Location is Exact	98109	47.641186	-122.342085	12

Figure 22 – Jupyter Notebook snapshot of the DataFrame (head and tail) containing the dimension Host. This DataFrame originates dimLocation.csv.

5.2.6. Process: the creation of Amenities outrigger (dimAmenities) and Facilities dimension (dimFacilities)

Input sources of Amenities and Facilities dimension: listings.csv

The output of Amenities and Facilities dimension: dimAmenities.csv and dimFacilities.csv

The code begins with the creation of the Amenities outrigger based on a one-hot encoded DataFrame that represents the presence or absence of each amenity. An auto-incrementing "amenitiesPK" column is inserted at the beginning of the DataFrame that serves as the primary key.

Next, we create the function "map_amenities" which adds a new column called "amenitiesFK" to store the mapped primary key value of amenities combination from the amenities dimension outrigger. This function is to be applied to each row in the "dimFacilities"

Then, the code proceeds to create the Facilities dimension “dimFacilities” by selecting specific columns from the "listings" DataFrame that contain information about the property's facilities, such as "property_type," "room_type," "accommodates," "bathrooms," "bedrooms," "beds," "bed_type," "guests_included," and "amenities". The columns in the

"dimFacilities" DataFrame are renamed to ensure clarity and consistency. Duplicate rows were then removed from the "dimFacilities" DataFrame, ensuring each combination of facility attributes is represented uniquely in the dimension table. An auto-incremental column called "facilitiesPK" is added to the DataFrame, assigns a unique identifier to each combination of facility attributes, and serves as the primary key for the "dimFacilities" table.

Figure 23 shows the DataFrame that originates dimAmenities.csv and dimFacilites.csv.

A

	amenitiesPK	alldaycheckin	airConditioning	breakfast	buzzerwirelessIntercom	cableTv	carbonMonoxideDetector	cat	dog	doorman	...	safetyCard	s
0	1	0	1	0	0	1	0	0	0	0	...	0	
1	2	1	0	0	1	1	1	0	0	0	...	1	
2	3	0	0	0	1	0	0	0	0	0	...	0	
3	4	0	0	0	1	0	0	0	0	0	...	0	
4	5	1	0	0	0	0	1	0	0	0	...	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
3109	3110	0	0	0	0	1	1	0	0	0	...	0	
3110	3111	0	0	0	0	0	1	0	0	0	...	1	
3111	3112	0	0	1	0	0	0	0	1	0	...	0	
3112	3113	0	0	0	0	0	1	1	0	0	...	1	
3113	3114	0	0	0	0	0	0	0	0	0	...	0	

B

	facilitiesPK	propertyType	roomType	accommodates	bathrooms	bedrooms	beds	bedType	guestsIncluded	amenitiesFK
0	1	Apartment	Entire home/apt	4	1.0	1.0	1.0	Real Bed	2	1
1	2	Apartment	Entire home/apt	5	2.0	2.0	2.0	Real Bed	4	2
2	3	Apartment	Entire home/apt	4	1.0	1.0	2.0	Real Bed	1	3619
3	4	Apartment	Private room	1	1.0	1.0	1.0	Real Bed	1	3
4	5	Apartment	Shared room	2	1.0	1.0	1.0	Real Bed	1	4
...	...	...	...	...	...	...	...	...	...	...
3131	3132	Townhouse	Entire home/apt	6	1.5	2.0	2.0	Real Bed	4	3109
3132	3133	Townhouse	Entire home/apt	5	2.5	2.0	2.0	Real Bed	4	1639
3133	3134	Townhouse	Private room	2	1.0	1.0	1.0	Real Bed	1	3111
3134	3135	Treehouse	Private room	2	1.0	1.0	1.0	Real Bed	1	3112
3135	3136	Yurt	Entire home/apt	3	1.0	0.0	2.0	Real Bed	2	3113

Figure 23 – Jupyter Notebook snapshot of the DataFrame (head and tail) containing (A) the amenities dimension and (B) the Facilities dimension. These DataFrames originate, respectively, dimAmenities and dimFacilities.csv. In (A), only the first columns are shown.

5.2.7. Process: the creation of the Price & Booking dimension

Input sources of Price & Booking dimension: listings.csv

The output of Price & Booking dimension: dimPriceAndBooking.csv

The code selects specific columns, 'instant_bookable' and 'cancellation_policy', from the "listings" DataFrame and creates a new DataFrame called "dimPriceAndBooking". The column names in "dimPriceAndBooking" are then renamed to 'instantBookable' and 'cancellationPolicy'. A list of price ranges is defined. Using the "instantBookable",

"cancellationPolicy", and "price_ranges" columns, all possible combinations are generated and stored in the "combinations" variable. An auto-incremental key column, 'nightPricePK', is added to "dimPriceAndBooking", serving as the primary key.

Figure 24 shows the DataFrame that originates dimPriceAndBooking.csv.

	nightPricePK	instantBookable	cancellationPolicy	nightPriceRange
0	1	0	moderate	0-49.99
1	2	0	moderate	50-99.99
2	3	0	moderate	100-149.99
3	4	0	moderate	150-199.99
4	5	0	moderate	200-299.99

Figure 24 – Jupyter Notebook snapshot of the DataFrame (head only) containing the dimension Price & Booking. This DataFrame originates dimPriceAndBooking.csv.

5.2.8. Process: the creation of the Price & Booking dimension

Input sources of Weather dimension: weather.csv

The output of Price & Booking dimension: dimWeather.csv

The code defines the possible values for each column in Fahrenheit temperature bands, precipitation bands, and rain status. The possible values for the "tMinBand" and "tMaxBand" columns are temperature ranges, while the "precipitationBand" column represents precipitation levels. The "rain" column has two values indicating whether it rained or not. Using these defined values, all possible combinations are generated and stored in the "combinations" variable. A new DataFrame called "dimWeather" is created using the combinations, with columns 'tMinBand', 'tMaxBand', 'precipitationBand', and 'rain'. An auto-incremental column "weatherPK" is added to the "dimWeather" DataFrame, assigning a unique identifier to each row and serving as the primary key.

Figure 25 shows the DataFrame that originates dimWeather.csv.

	weatherPK	tMinBand	tMaxBand	precipitationBand	rain
0	1	>=14	>=14	0-0.99	It rained
1	2	>=14	>=14	0-0.99	It did not rain
2	3	>=14	>=14	1-2.99	It rained
3	4	>=14	>=14	1-2.99	It did not rain
4	5	>=14	>=14	3-4.99	It rained
...	...	...	...	...	...
283	284	>=87	>=87	1-2.99	It did not rain
284	285	>=87	>=87	3-4.99	It rained
285	286	>=87	>=87	3-4.99	It did not rain
286	287	>=87	>=87	>=5	It rained
287	288	>=87	>=87	>=5	It did not rain

Figure 25 – Jupyter Notebook snapshot of the DataFrame (head and tail) containing the dimension Weather. This DataFrame originates dimPriceAndBooking.csv.

5.2.9. Process: the creation of Customer dimension

Input sources of Customer dimension: reviews.csv

The output of Customer dimension: dimCustomer.csv

The code selects the 'reviewer_id' and 'reviewer_name' columns from the "reviews" DataFrame and creates a new DataFrame called "dimCustomer" with these columns. The column names in the "dimCustomer" DataFrame is then renamed to 'reviewerID' and 'reviewerName'. Duplicate rows in the DataFrame are dropped based on all columns. The DataFrame is sorted in ascending order based on the 'reviewerID' column. An auto-incremental column called "reviewerPK" is added to the "dimCustomer" DataFrame, assigning a unique identifier to each row and serving as a primary key.

Figure 26 shows the DataFrame that originates dimCustomer.csv.

	reviewerPK	reviewerID	reviewerName
0	1	15	Kat
1	2	262	Jonathan
2	3	431	Matthew
3	4	1618	Elaine
4	5	1720	Ryan
...	...	...	...
75712	75713	52713564	Mary
75713	75714	52721080	Ashley
75714	75715	52723379	Xavi
75715	75716	52790726	Paul
75716	75717	52812740	Arin

Figure 26 – Jupyter Notebook snapshot of the DataFrame (head and tail) containing the dimension Customer. This DataFrame originates dimCustomer.csv.

5.2.10. Process: the creation of StaysFact

Input sources of StayFact: reviews.csv and all the dimensions csvs created above.

The output of Customer dimension: staysFact.csv

The code creates the fact table by creating the columns to store the keys corresponding to each dimension table and business process measures. The business process measure is the stay evaluation, obtained by sentiment analysis of the reviews. It would be interesting to have additional metrics such as the total number of nights so we could calculate the stay value. However, since this information is not available, the only metric we could calculate was the stay evaluation.

	stayPK	dateFK	propertyFK	nightPriceFK	customerFK	streetFK	hostFK	facilitiesFK	weatherFK	stayEvaluation	count
0	1	1761	2373	32	57573	3196	2346	591	186	0.433333	1
1	2	1762	2373	32	62071	3196	2346	591	178	0.301136	1
2	3	1768	2373	32	68032	3196	2346	591	177	0.410417	1
3	4	1775	2373	32	63614	3196	2346	591	186	0.358333	1
4	5	1783	2373	32	65184	3196	2346	591	178	0.493485	1
...	...	...	...	...	...	...	...	...	...	...	...
84826	84827	1846	1050	28	67769	1567	1451	91	169	0.194844	1
84827	84828	1851	1050	28	23123	1567	1451	91	169	0.311111	1
84828	84829	1854	1050	28	45300	1567	1451	91	170	0.000000	1
84829	84830	1867	1050	28	51889	1567	1451	91	121	0.216852	1
84830	84831	1908	3623	2	60061	3593	1833	298	113	0.367302	1

Figure 27 – Jupyter Notebook snapshot of the DataFrame (head and tail) containing the fact table. This DataFrame originates staysFact.csv.

5.3. Analytical reports

After extraction, transformation, and loading, we proceeded with the analytical reports phase. We used PowerBI to better visualize the results of our queries and try to answer the analytical questions defined in the first stage.

Overall, this process resulted in the dimension table, fact table and data cube represented in **Figure 28.**

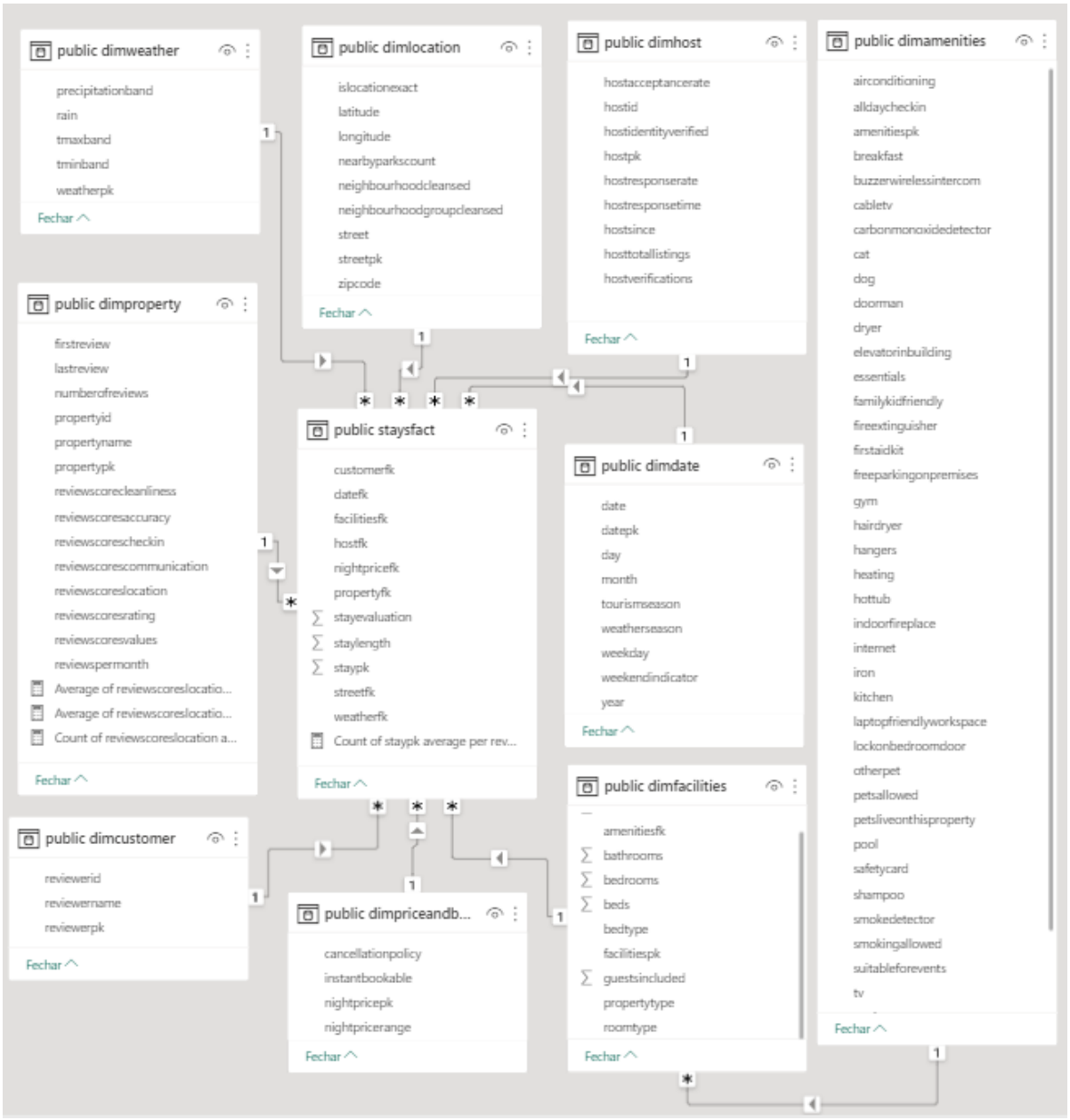


Figure 28 – PowerBI snapshot of the data cube generated in this project.

5.3.1. Analytical question 1

What factors are associated with higher/lower prices for Airbnb listings? For example, does having more bedrooms, better ratings, or a certain location increase the price of a listing?

To answer this question, we performed the analytical report analysis shown in **Figure 29**.

For this analysis we found that most bookings concentrate in the price range of 50-99.99, followed by the price range of 100-149.99 (**Figure 29A**). This constitutes almost 2/3 of the stays and shows that low price is not the main driver of choice. Therefore, it is worth

considering which additional factors might be driving pricing and investing in these, since the low budget segment is clearly not in high demand in Seattle.

Downtown and Capitol Hill are clearly the preferred locations for stay and they are also the neighborhoods where higher prices can be charged (**Figure 29B**). Together with Queen Anne, these neighborhoods concentrate most properties where the highest price tags could be charged, therefore these are good locations to invest in if the client wants to operate in the premium segment.

The highest scores clearly concentrate in the price range of 50-99.99, followed by the price range of 100-149.99 (**Figure 29C**), This shows that pricing is a strong influence of the overall property appreciation. Therefore, if the client wants to charge higher prices, needs to search for additional factors that can compensate for this anchor and be able to match the high expectations of those who are willing to pay more.

One such factor seems to be the number of bedrooms (**Figure 29D**), where we see that very high prices concentrate on properties with 3 or more bedrooms. Most stays concentrate on one-bedroom properties, therefore, if the client has the possibility to invest in two small properties instead of a bigger one this will likely have more demand and be more profitable.

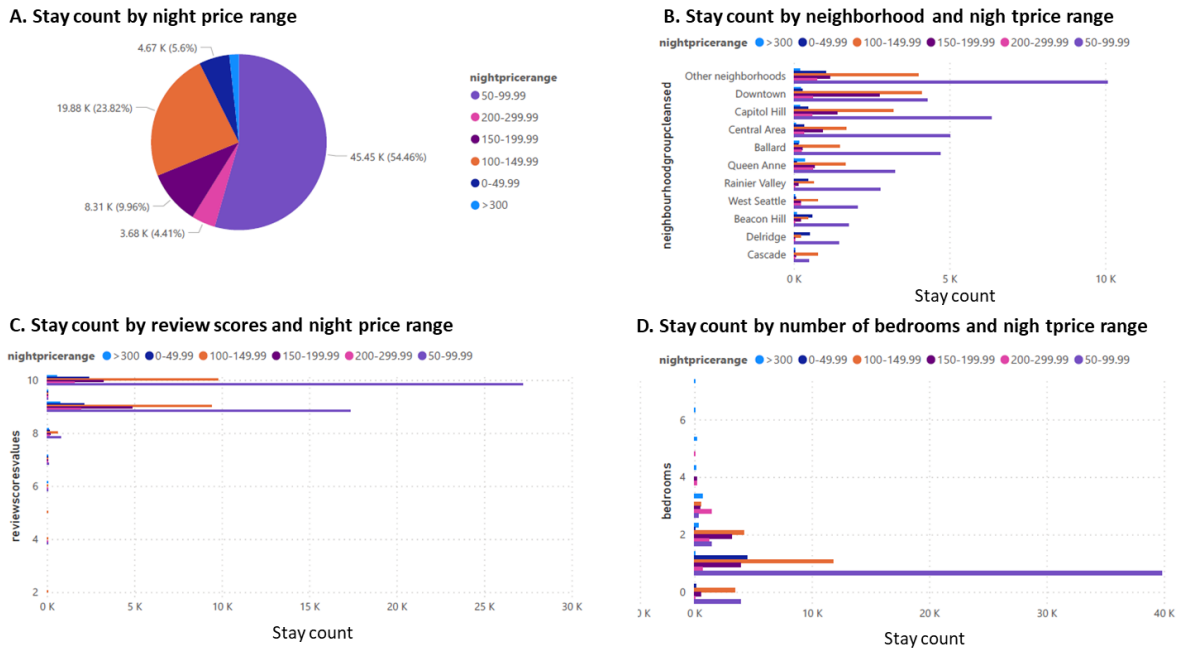


Figure 29 – Analytical reports on possible factors influencing nigh price and demand.

5.3.2. Analytical question 2

How are the properties located in Seattle? Are there specific neighborhoods where Airbnbs are more common?

This question was answered in stage one when analyzing the geographical distribution of the properties (Figure 10) where we identified a good distribution of the properties across the entire Seattle, with some concentration on downtown, Fremont, and the University District. This aligns with the analysis carried out in the context of analytical question 1, where, for example, we saw that downtown was one of the areas with more demand. Focusing on these areas will guarantee high demand.

5.3.3. Analytical question 3

How does the weather influence the number of reservations? And the weekday?

To answer this question, we performed the analytical report analysis shown in **Figure 30**.

This analysis showed that the demand is higher in summer and autumn (**Figure 30A**). Therefore, these are the months when higher prices can be charged. We also found that most stays concentrate during the week (**Figure 30B**). This shows that the Airbnb market in Seattle is focused not only on vacation periods, but also on other types of client profile, probably individuals visiting Seattle on business. This is corroborated by the evidence that the number of stays is well-balanced between the high and low tourism season (**Figure 30C**), and probably contributes to the high demand of one-bedroom properties that we saw in analytical question 1. This also shows that the common strategy of lowering the price in low tourism season is probably not necessary in Seattle

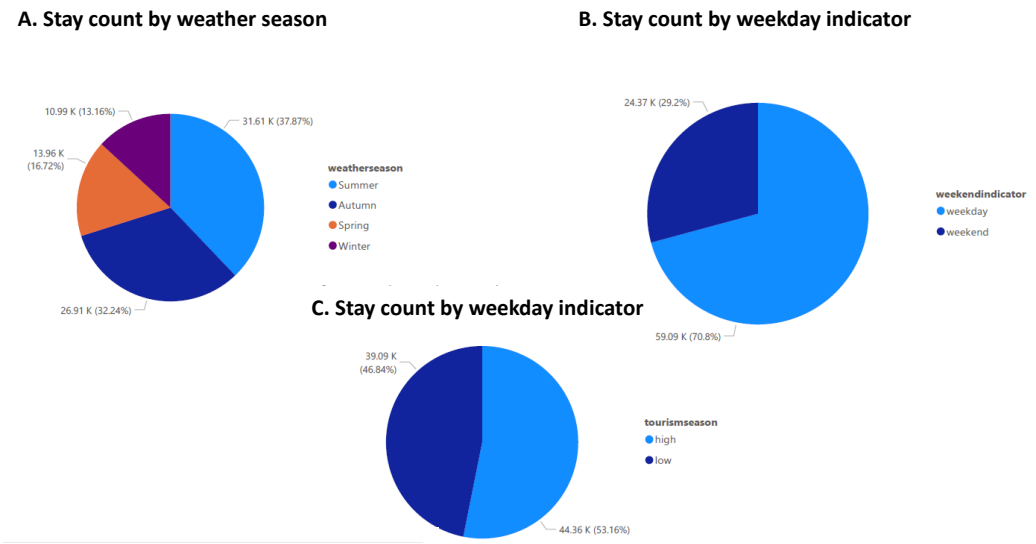


Figure 30 – Analytical reports on weather and weekday factors influencing demand.