2022

Big Data Architecture & Governance

Projected sales of main products in 2013
Distribution of market share among the major industry players
Share of market activity
25%
75%
Passive market share
Projected sales of main products in 2013

Northeastern University

Assignment Name:

Airbnb Ratings

Student Names:

Nandita Patil

Shweta Wakale

Siddharth Natekar

1. Contents

2. Assignment

2.1. Case

Each team should select a dataset to analyze and build an analytical dashboard as a Proof-of-concept to illustrate the value of data driven analytics. You need to present your dataset.

.

2.2. Assignment Goals

To work with datasets, Perform/Create:

- Create you group assignment project in Velero:
  - Project
  - Project Plan
  - Resource Allocation
  - Timesheet
  - Issues & Risks.
  - You are required to report on your team progress every week
- **Data Profiling** – Using Python profiling library, describe your understanding of the data.
- **Data Wrangling and Cleansing** - Pandas/Alteryx/XSV
  - Filtering and Aggregating if needed.
  - Missing value handling.
  - Deriving additional columns from existing datasets if needed.
  - Cleaning (removing blank spaces, formatting dates, Capitalizing etc.) .
- Database Installation: Install NEO4J database .
- Data Mapping and Integration to your Database for the Entire Dataset.
- **Business and Technical Metadata** – develop business term list describing all the data elements available in the file.
- **Data Validation** – Validate the data using python data libraries.
- **Data Visualization** – Create a presentation dashboard to reflect your understanding of the data, you may use python visualization libraries or Power BI
- **System Integration and User Acceptance Testing** - Test Cases – describe your validation & testing process.
- **Risks/Issues** – identify risks and issues related to your project.
- Describe challenges encountered and how you resolved them.
- **End User Instructions (Steps to run your Dashboard)** – provide a full description how to run your process:
  - Database Creation and load.

- ○ Visualization interpretation - describe information regarding your findings.

2.2.1. VISUALIZATION DELIVERABLES

Once you wrangle/clean/join/integrate the data, import the data into **NEO4J** and illustrate how to use the appropriate graph to illustrate various aspects of analysis.

Questions to consider:

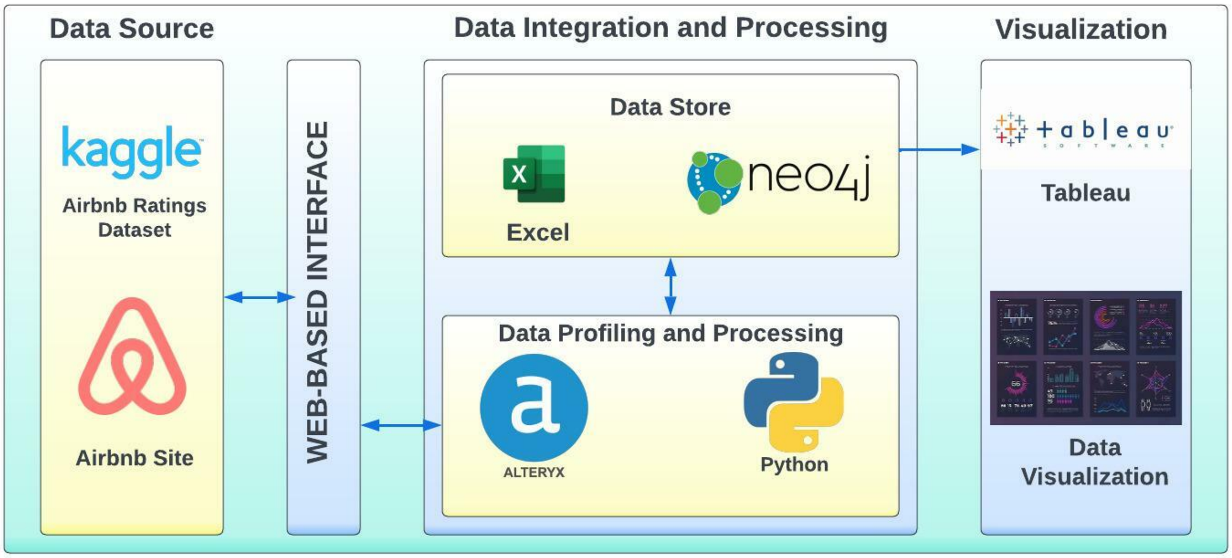
- Columns used for dimensions, and columns that are used for measurement.
- How would you generate new dimensions if needed
- Who would use this dashboard and how they benefit from your dashboard
- What value would be generated using this dashboard

2.2.2. OTHER DELIVERABLES

- Presentation of the entire work from the first step till the dashboards including the Velero screenshots.
- Business and technical metadata presentation – Identifying all available business terms and extracting related technical metadata.
- Complete explanation of the dashboard and usability.
- Complete instruction as how to implement and run the database load, technical meta data extraction, and dashboard.

3. Documentation

3.1. Vision Diagram



3.2. Data Wrangling and Cleansing

Description Of Data

The dataset includes four main tables:

- **Listings -** Detailed listings data about hosts, Airbnb Houses and Price. The attributes used in the analysis are:
  - › **Listing_ID:** Unique ID for listings
  - › **Name:** Name of the Listing
  - › **Property_type:** Types of property
  - › **Room_type:** Listing space type
  - › **Accomodation:** Number of people an Airbnb can Accommodate
  - › **Neighbourhood_cleaned**
  - › **Price:** Price of the Listing in Dollars
  - › **Amenities:** Amenities available in Airbnb
- **Host –** Detailed Host data**.** Key attributes used in the analysis are:
  - › **Host_ID:** Unique ID for Host
  - › **Host_Name:** Name of the Host
  - › **Host_Listing_Count:** The Total number of host listings
  - › **Host_Response_Rate:** Response Rate of the Host

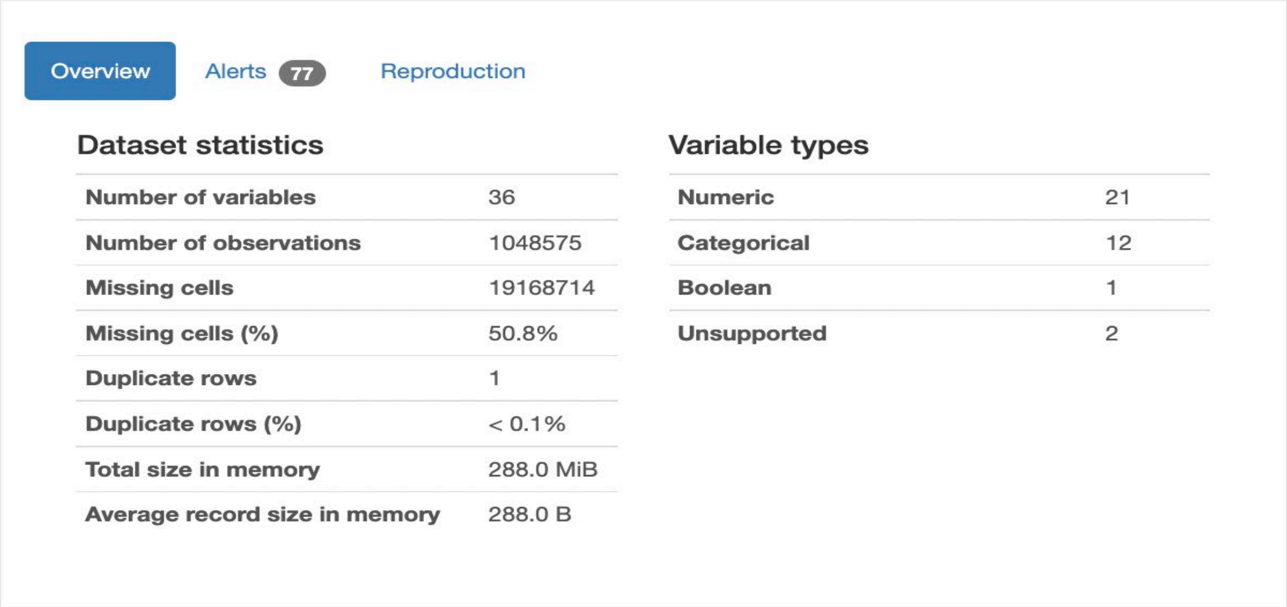
- **Reviews –** Detailed reviews given by the guests. Key attributes include:
  - **Review Scores Accuracy**: how accurately did the listing page represent an Airbnb
  - **Review Scores Cleanliness**: how clean and tidy did the guests feel about an Airbnb
  - **Review Scores Checkin**: how smoothly did check-in go
  - **Review Scores Communication**: how well did the guests communicate with the hosts before and during the stay
  - **Review Scores Location**: how did guests feel about the neighborhood
  - **Review Scores Value** : did the guest feel that the listing provided good value for the price.
  - **Number of reviews**: the total number of reviews
- **Address –** Provides details about Airbnb location in New York City. The attributes includes:
  - **Longitude:** The longitude of the Airbnb
  - **Latitude:** The latitude of the Airbnb
  - **Country:** Country where the Airbnb is located
  - **State:** State where the Airbnb is located
  - **City:** City where the Airbnb is located

Data Profiling:

The report generated by Pandas profiling is **a complete analysis without any input from the user except the dataframe object**. All the elements of the report are chosen automatically, and default values are preferred.

Pandas Profiling on Airbnb Ratings Dataset:

Overview



Dataset statistics

Number of variables	36
Number of observations	1048575
Missing cells	19168714
Missing cells (%)	50.8%
Duplicate rows	1
Duplicate rows (%)	< 0.1%
Total size in memory	288.0 MiB
Average record size in memory	288.0 B

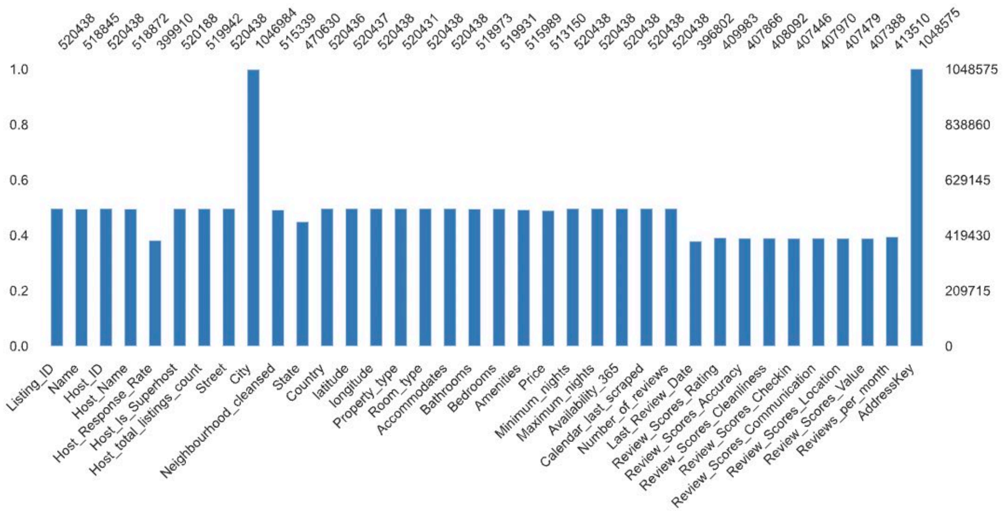
Variable types

Numeric	21
Categorical	12
Boolean	1
Unsupported	2

From the overview, we can analyze that the dataset has **1048575** observations
The dataset has **36** variables
The Dataset has,
12 Categorical values
2 Unsupported
21 Numerical values
1 Boolean value

Missing values

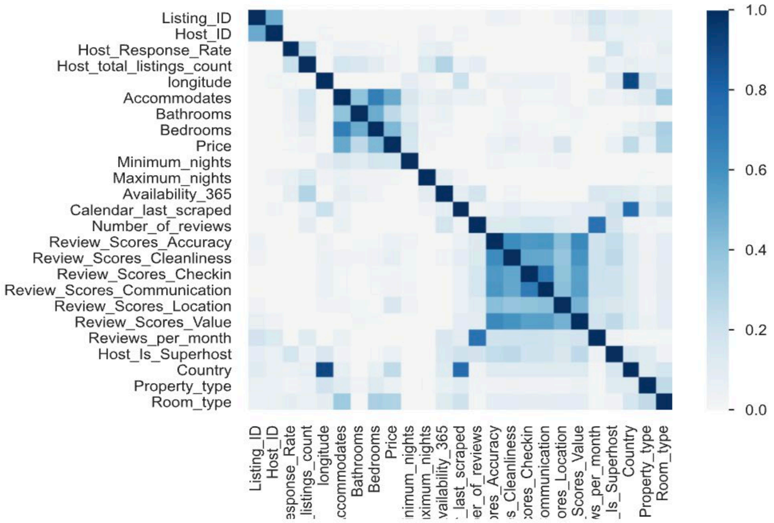
Count Matrix Heatmap Dendrogram

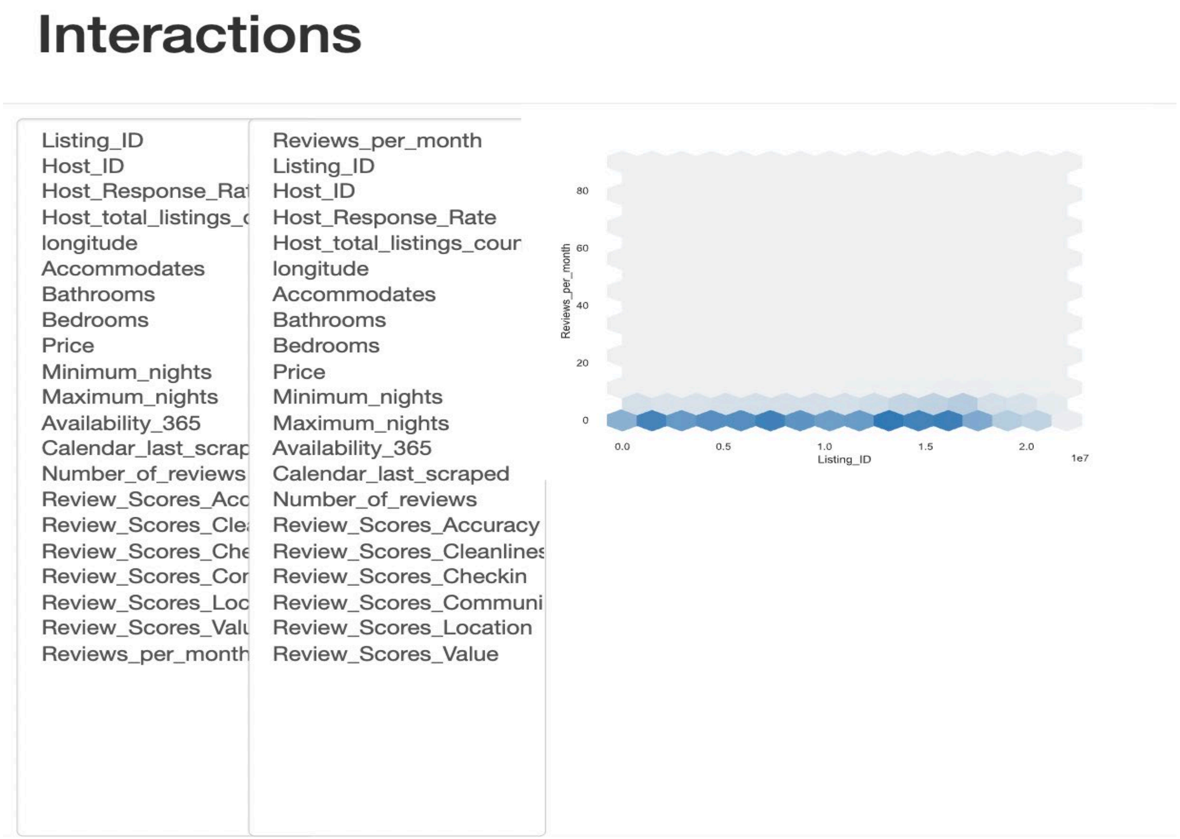


A simple visualization of nullity by column.

Correlations

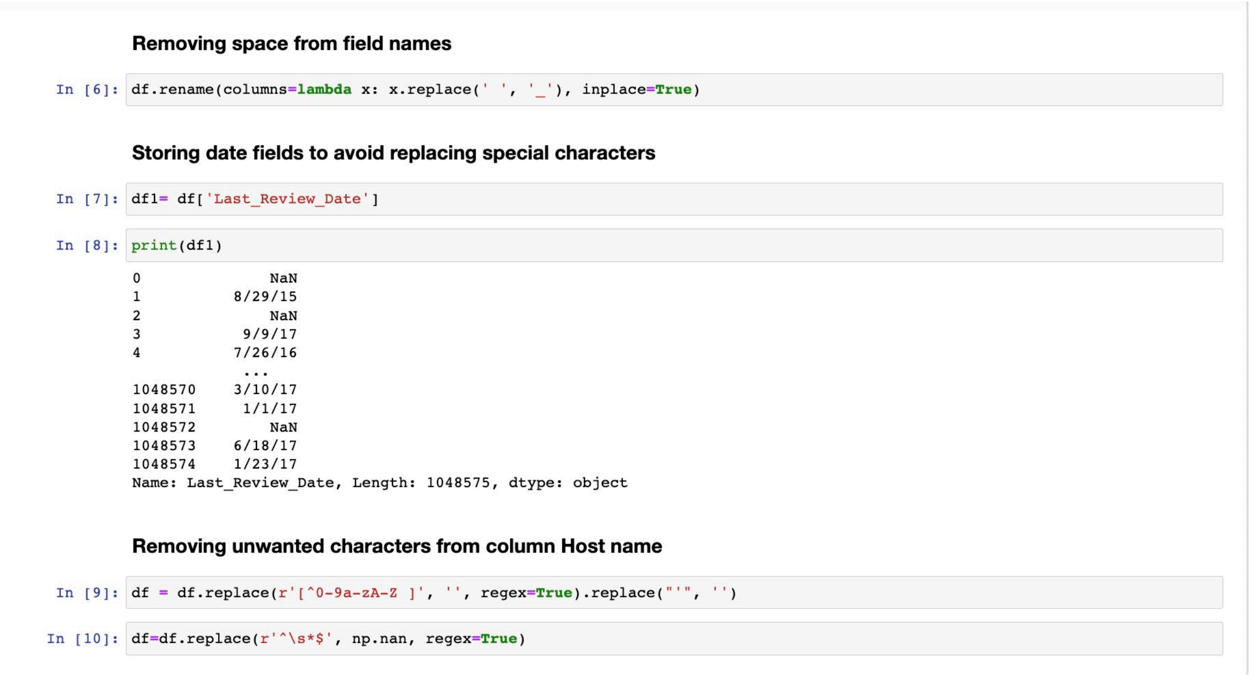
Auto Spearman's ρ Pearson's r Kendall's τ Cramér's V (φc) Toggle correlation descriptions Phik (φk)





Interactions

Data Preprocessing



Removing space from field names

```
In [6]: df.rename(columns=lambda x: x.replace(' ', '_'), inplace=True)
```

Storing date fields to avoid replacing special characters

```
In [7]: df1= df['Last_Review_Date']
```

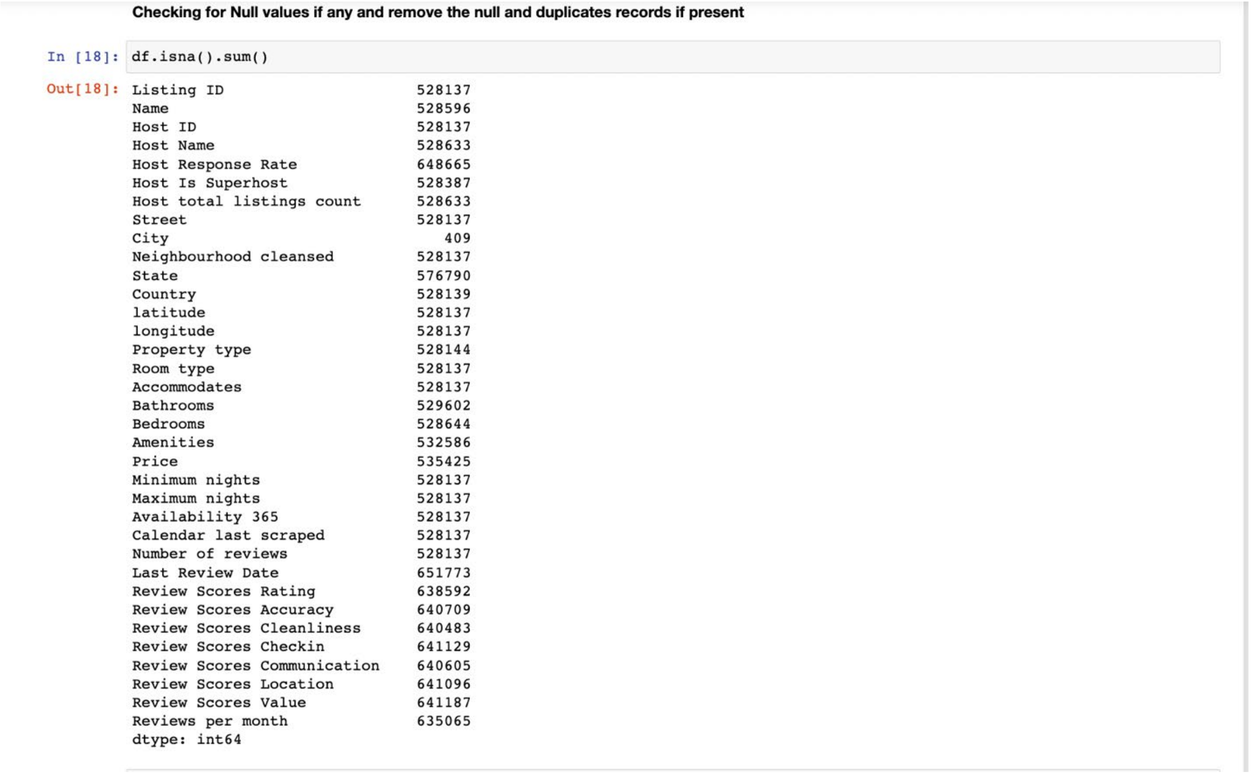
```
In [8]: print(df1)

0                NaN
1            8/29/15
2                NaN
3             9/9/17
4            7/26/16
              ...
1048570      3/10/17
1048571       1/1/17
1048572          NaN
1048573      6/18/17
1048574      1/23/17
Name: Last_Review_Date, Length: 1048575, dtype: object
```

Removing unwanted characters from column Host name

```
In [9]: df = df.replace(r'[^0-9a-zA-Z ]', '', regex=True).replace("'", '')
```

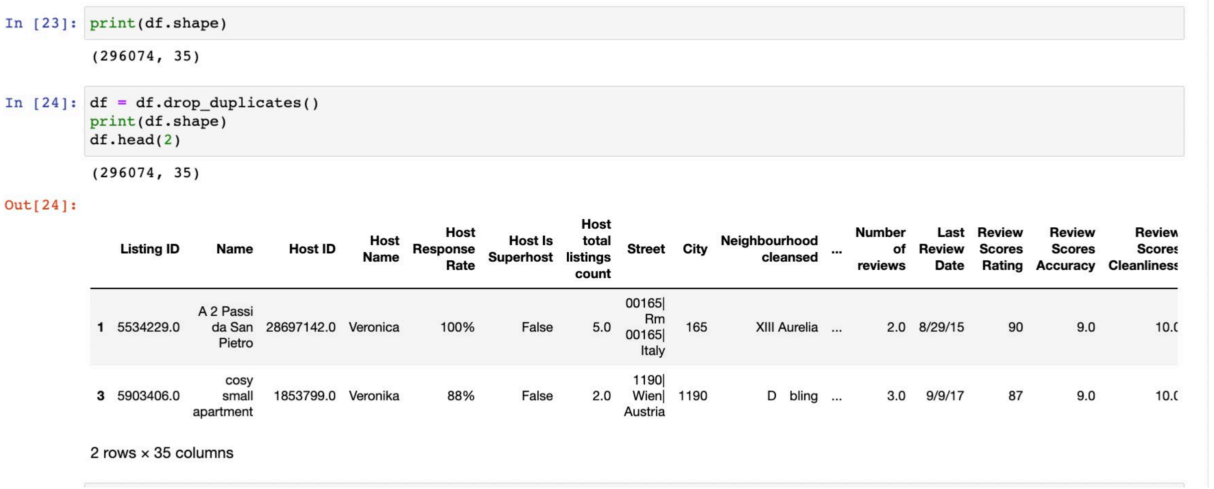
```
In [10]: df=df.replace(r'^\s*$', np.nan, regex=True)
```

Checking and Removing Null and Duplicate values



Checking for Null values if any and remove the null and duplicates records if present

```
In [18]: df.isna().sum()

Out[18]: Listing ID                      528137
         Name                            528596
         Host ID                         528137
         Host Name                       528633
         Host Response Rate              648665
         Host Is Superhost               528387
         Host total listings count       528633
         Street                          528137
         City                               409
         Neighbourhood cleansed          528137
         State                           576790
         Country                         528139
         latitude                        528137
         longitude                       528137
         Property type                   528144
         Room type                       528137
         Accommodates                    528137
         Bathrooms                       529602
         Bedrooms                        528644
         Amenities                       532586
         Price                           535425
         Minimum nights                  528137
         Maximum nights                  528137
         Availability 365                528137
         Calendar last scraped           528137
         Number of reviews               528137
         Last Review Date                651773
         Review Scores Rating            638592
         Review Scores Accuracy          640709
         Review Scores Cleanliness       640483
         Review Scores Checkin           641129
         Review Scores Communication     640605
         Review Scores Location          641096
         Review Scores Value             641187
         Reviews per month               635065
         dtype: int64
```



In [23]:
```
print(df.shape)
```
```
(296074, 35)
```

In [24]:
```
df = df.drop_duplicates()
print(df.shape)
df.head(2)
```
```
(296074, 35)
```

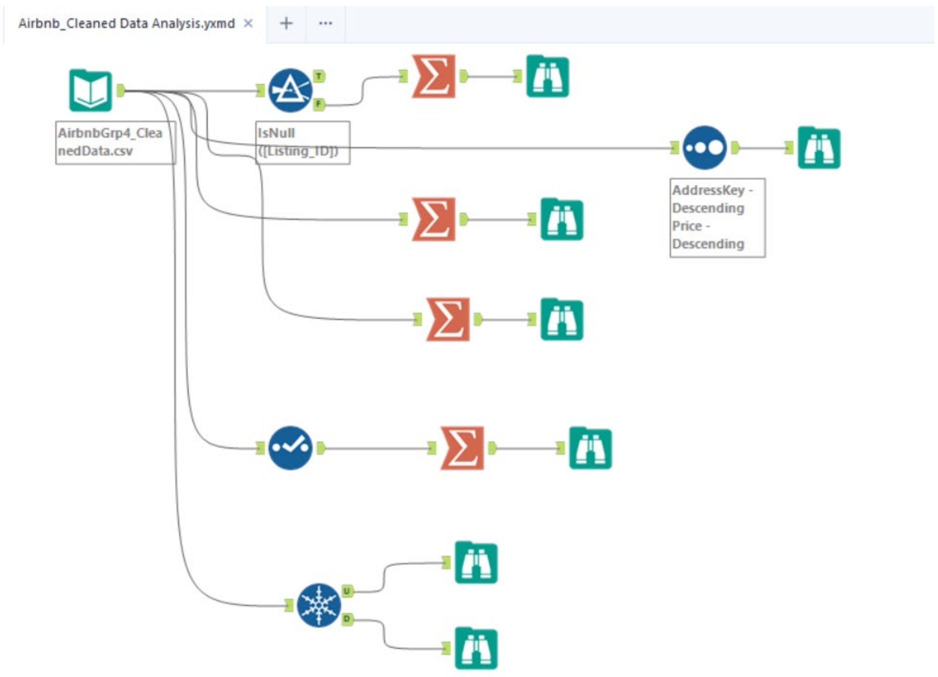
Out[24]:

	Listing ID	Name	Host ID	Host Name	Host Response Rate	Host Is Superhost	Host total listings count	Street	City	Neighbourhood cleansed	...	Number of reviews	Last Review Date	Review Scores Rating	Review Scores Accuracy	Review Scores Cleanliness
1	5534229.0	A 2 Passi da San Pietro	28697142.0	Veronica	100%	False	5.0	00165| Rm 00165| Italy	165	XIII Aurelia	...	2.0	8/29/15	90	9.0	10.0
3	5903406.0	cosy small apartment	1853799.0	Veronika	88%	False	2.0	1190| Wien| Austria	1190	D bling	...	3.0	9/9/17	87	9.0	10.0

2 rows × 35 columns

Alteryx

Alteryx is designed to make advanced analytics accessible to any data worker. We used Alteryx for Data Preprocessing.
Alteryx Workflow for Cleaning the dataset which includes removing blank spaces, extra commas, and Capitalization.


Airbnb_Cleaned Data Analysis.yxmd
AirbnbGrp4_CleanedData.csv
IsNull ([Listing_ID])
T
F
AddressKey - Descending Price - Descending
U
D

Data Profiling done using cleaned to cross validate values populating in the Tableau views Total record count, country wise record counts, null cross-check if generated during the run, summarized the dimensions to check the categorical record count.

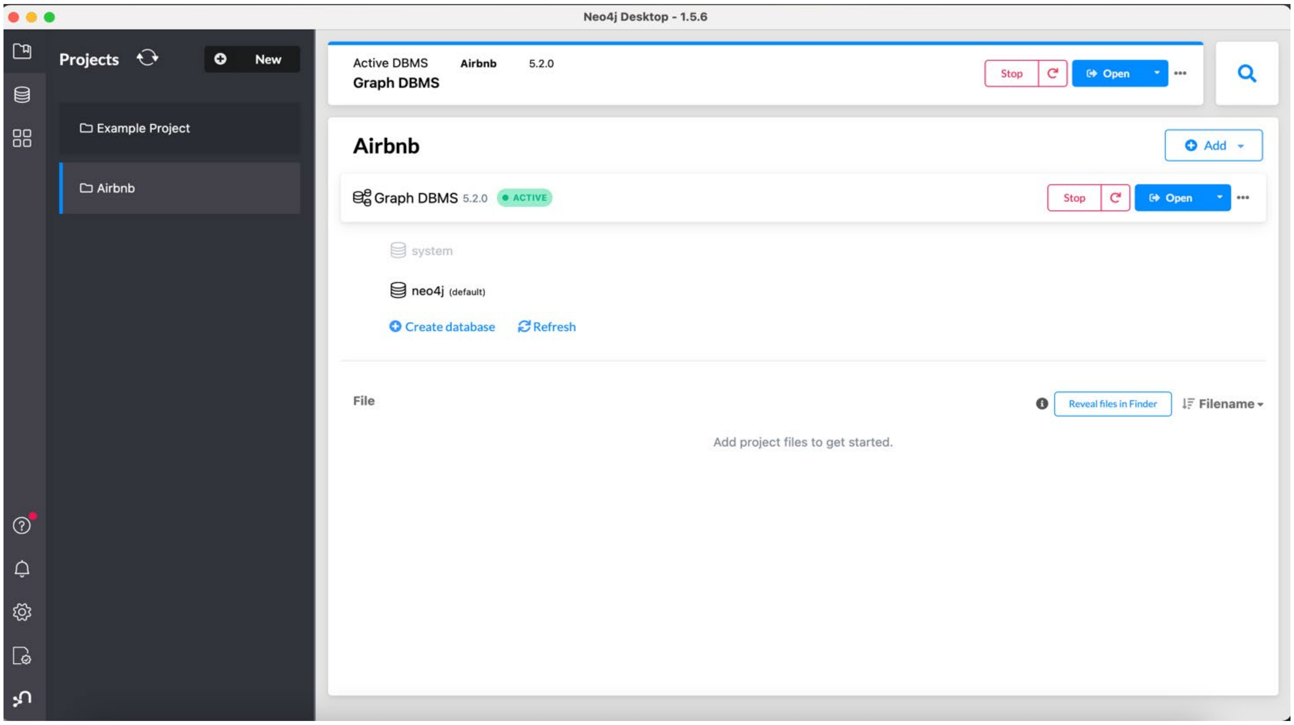
3.3. Database Installation

Neo4j Database

The Neo4j database is a graph database and is used to represent the data in the form of graphs. It offers data integrity and is ACID (Atomic, Consistent,Isolated,Durable) compliant. Just like RDBMS has a language called SQL to access data, the Graph database has a language called Cypher
Neo4j can be download by clicking on the link **Download Neo4j**

Starting the Server using Neo4j Desktop

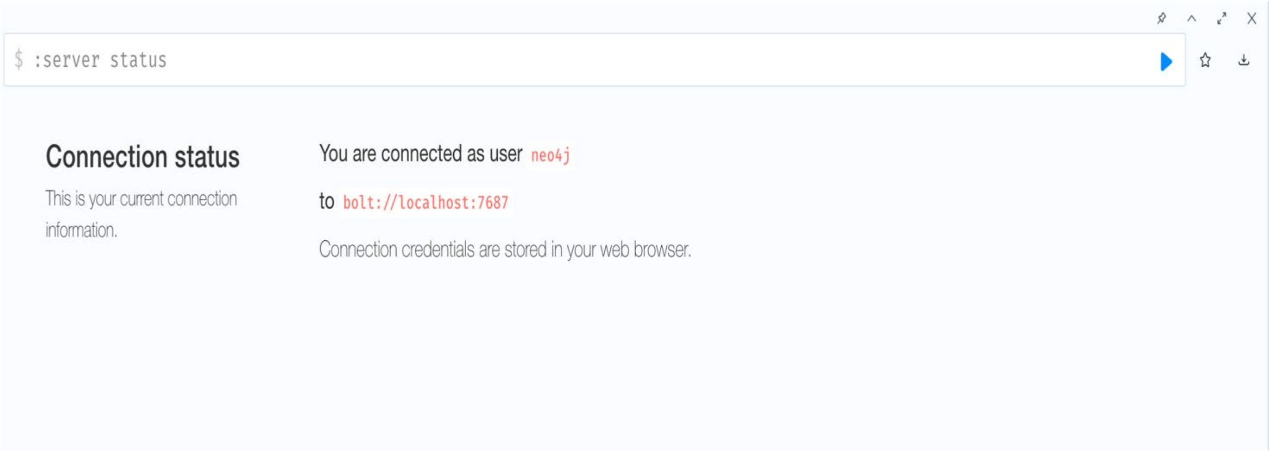
To start using **Neo4j database**, open the Neo4j Desktop installed on the system.
Click on **NEW** to create a Project.
Now, create a database by clicking the **“Add”** button and set a password for your database. It is also possible to change the password by going into the administration tab and setting a new password. Add all the relevant files related to the database in the files section.



Add project files to get started.

Start the server by clicking the play button on the window, stop the server or restart the server when the database is not needed. Now, the database is ready and query the data using the Neo4j
Browser **“Open browser”** or through command line **“Open Terminal”**.

You can also open a new window in your preferred browser and type
http://www.localhost:7687
entered for your database and click **“Connect”**.
into the URL. To connect, you will need to enter the password you



```
$ :server status
```

Connection status

This is your current connection information.

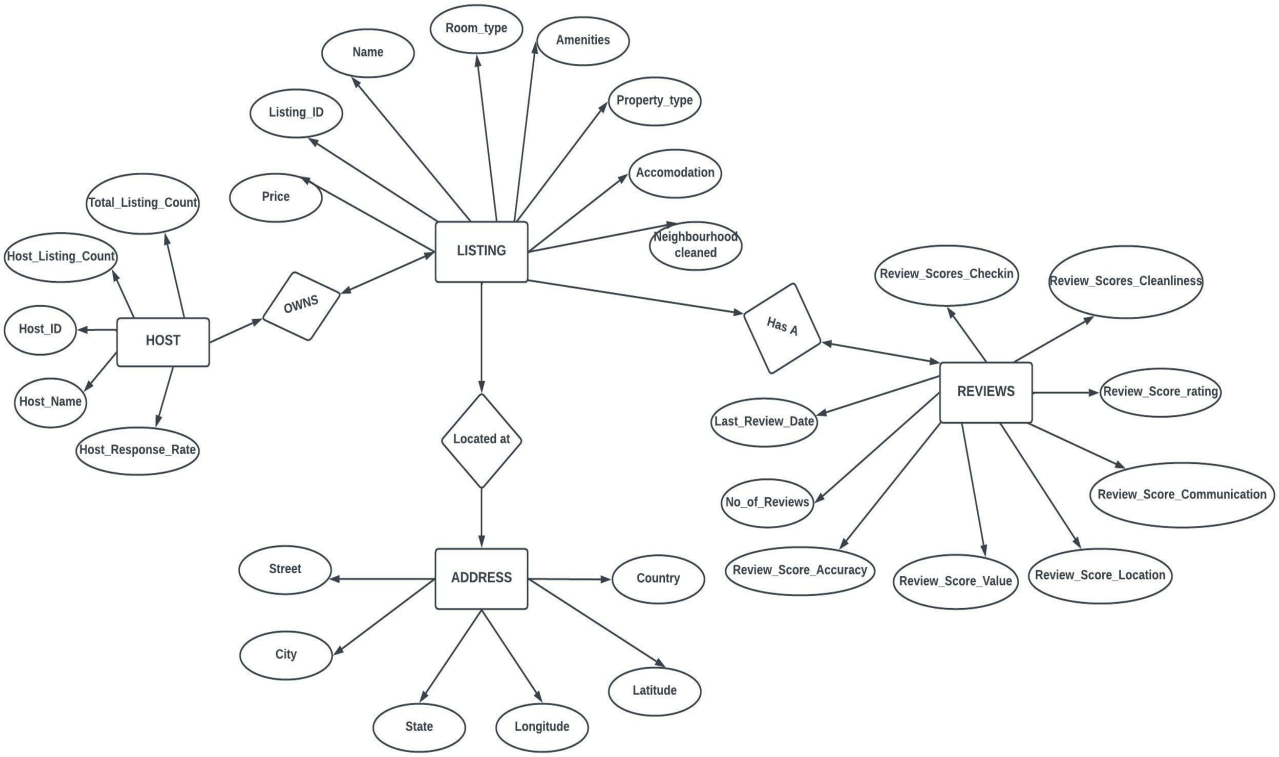
You are connected as user neo4j

to bolt://localhost:7687

Connection credentials are stored in your web browser.

3.4. Data Mapping and Integration

DATA MODEL



CYPHER QUERY FOR LOADING AIRBNB RATINGS DATASET IN NEO4J DATABASE

# Defining Constraints for Database

```
CREATE CONSTRAINT ON (host:Host) ASSERT host.HostId IS UNIQUE;
CREATE CONSTRAINT ON (listing:Listing) ASSERT listing.ListingId IS UNIQUE;
CREATE CONSTRAINT ON (address:Address) ASSERT address.AddressKey IS UNIQUE;
CREATE CONSTRAINT Reviews
FOR (reviews:Reviews)
REQUIRE (reviews.Last_Review_Date) IS NODE KEY
```

#Creating HOST Node

```
:auto USING PERIODIC COMMIT 500
LOAD CSV With HEADERS FROM 'file:///AirbnbGrp4_CleanedData.csv' AS row

MERGE(host:Host{HostId:row.Host_ID})
ON CREATE SET
host.HostName=row.Host_Name,
host.HostResponseRate=row.Host_Response_Rate,
host.HostIsSuperhost=row.Host_Is_Superhost,
host.HostTotalListingsCount= row.Host_total_listings_count;
```

#Creating LISTINGS Node

```
:auto USING PERIODIC COMMIT 500
LOAD CSV With HEADERS FROM 'file:///AirbnbGrp4_CleanedData.csv' AS row
MERGE (listing:Listing{ListingId:row.Listing_ID})
ON CREATE SET
listing.Name=row.Name,
 listing.NeighbourhoodCleansed=row.Neighbourhood_cleansed,
listing.Propertytype=row.Property_type,
listing.RoomType=row.Room_type, listing.Accommodates=row.Accommodates,
listing.Bathrooms=row.Bathrooms,listing.Bedrooms=row.Bedrooms,
listing.Amenities=row.Amenities,
listing.Price=row.Price,
listing.MinimumNights=row.Minimum_nights,
listing.MaximumNights=row.Maximum_nights,
listing.Availability365=row.Availability_365;
```

#Creating ADDRESS Node

```
:auto USING PERIODIC COMMIT 500
LOAD CSV With HEADERS FROM 'file:///AirbnbGrp4_CleanedData.csv' AS row
MERGE
(address:Address{AddressKey:row.AddressKey})
ON CREATE SET
address.Street=row.Street,address.City=row.City,
address.State=row.State,
address.Country= row.Country,
address.Longitude= row.Longitude,
 address.Latitude= row.Latitude;
```

# Creating REVIEWS Node

```
:auto USING PERIODIC COMMIT 500
LOAD CSV With HEADERS FROM 'file:///AirbnbGrp4_CleanedData.csv' AS row
MERGE (reviews:Reviews{Last_Review_Date:row.Last_Review_Date})
ON CREATE SET
reviews.Review_Scores_Rating=row.Review_Scores_Rating,
reviews.Review_Scores_Accuracy=row.Review_Scores_Accuracy,
reviews.Review_Scores_Cleanliness=row.Review_Scores_Cleanliness,
reviews.Review_Scores_Checkin=row.Review_Scores_Checkin,
reviews.Review_Scores_Communication=row.Review_Scores_Communication,
reviews.Review_Scores_Location=row.Review_Scores_Location,
reviews.Review_Scores_Value=row.Review_Scores_Value,
reviews.Review_Scores_month=row.Review_Scores_month;
```

# Creating relationship between HOST to LISTING

```
LOAD CSV WITH HEADERS FROM "file:///AirbnbGrp4_CleanedData.csv" AS row
MATCH (host:Host{HostId:row.Host_ID})
MATCH (listing:Listing {ListingId:row.Listing_ID})
MERGE (host)-[:OWNS]->(listing);
```

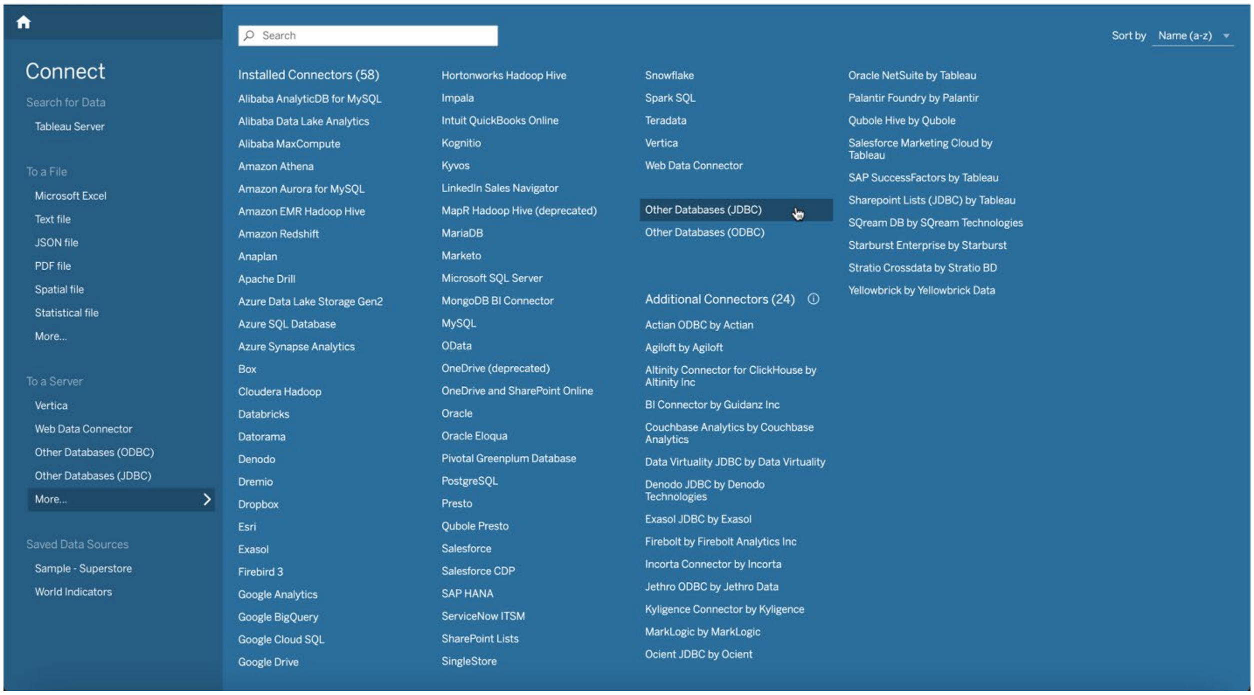
# Creating Relationship between LISTING to REVIEWS

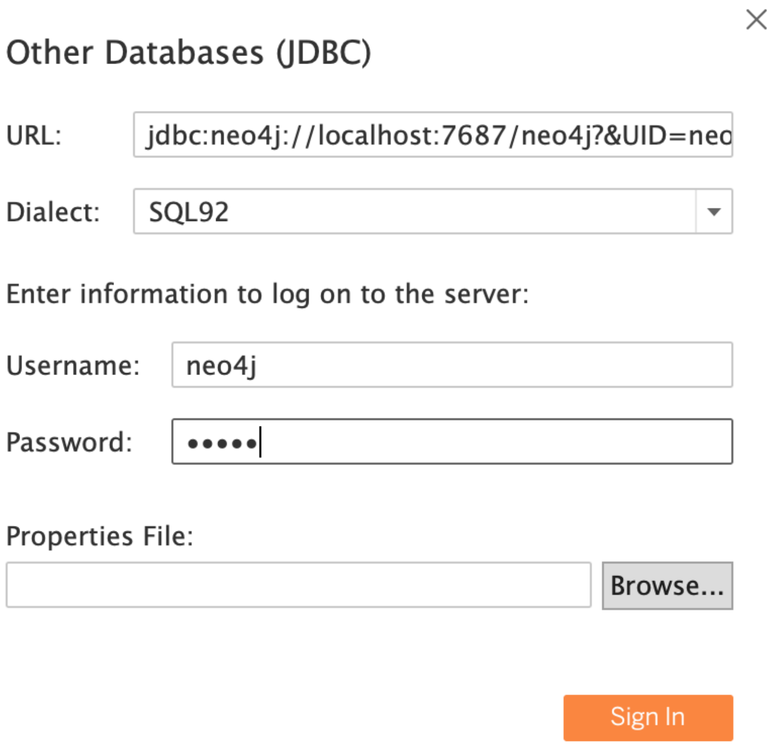
```
LOAD CSV WITH HEADERS FROM "file:///AirbnbGrp4_CleanedData.csv" AS row
MATCH (reviews:Reviews{Last_Review_Date:row.Last_Review_Date})
MATCH (listing:Listing {ListingId:row.Listing_ID})
MERGE (listing)-[:HASA]->(reviews);
```

# Creating Relationship between LISTING to ADDRESS

```
LOAD CSV WITH HEADERS FROM "file:///AirbnbGrp4_CleanedData.csv" AS row
MATCH (address:Address{AddressKey:row.AddressKey})
MATCH (listing:Listing {ListingId:row.Listing_ID})
MERGE (listing)-[:LOCATEDAT]->(address);
```

Connecting Neo4j to Tableau Using JDBC Connector for Data Visualization




Other Databases (JDBC)

URL: jdbc:neo4j://localhost:7687/neo4j?&UID=neo

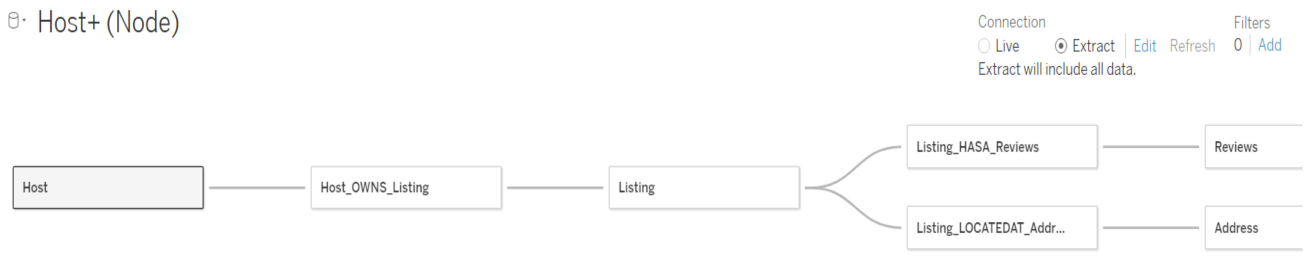
Dialect: SQL92

Enter information to log on to the server:

Username: neo4j

Password: •••••

Properties File:



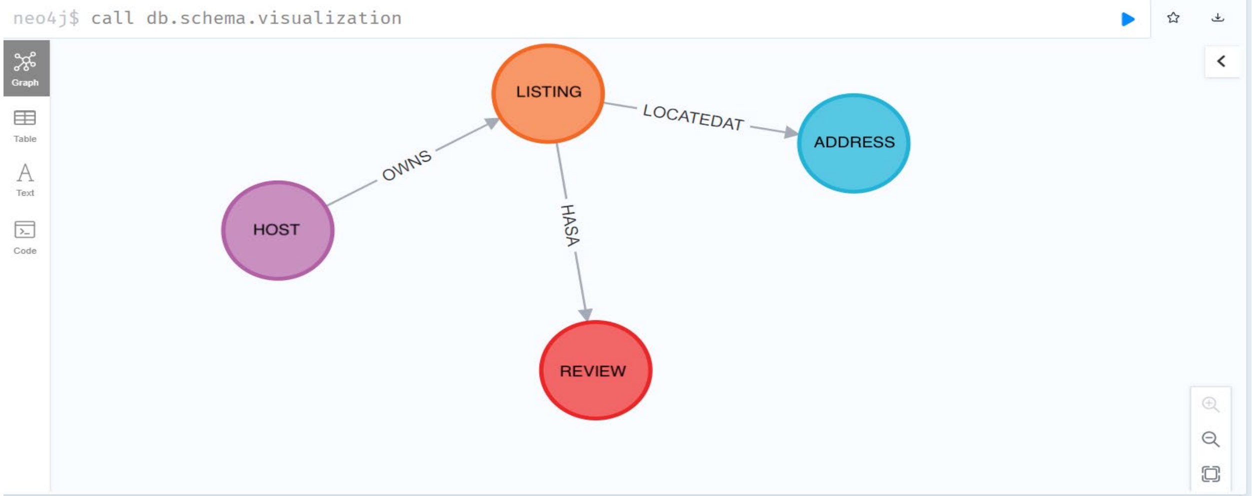
Host+ (Node)

Connection
○ Live ● Extract | Edit Refresh

Filters
0 | Add

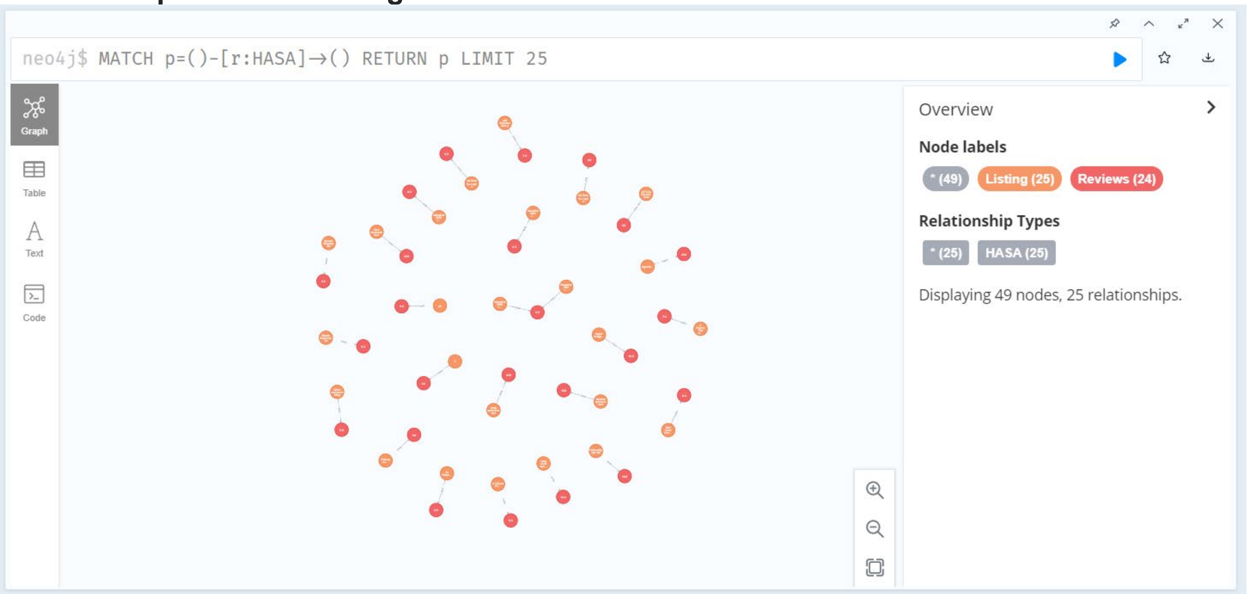
Extract will include all data.

Database Schema in Neo4J



```
neo4j$ call db.schema.visualization
```

Relationship between Listings and Reviews



```
neo4j$ MATCH p=()-[r:HASA]→() RETURN p LIMIT 25
```

Overview

Node labels

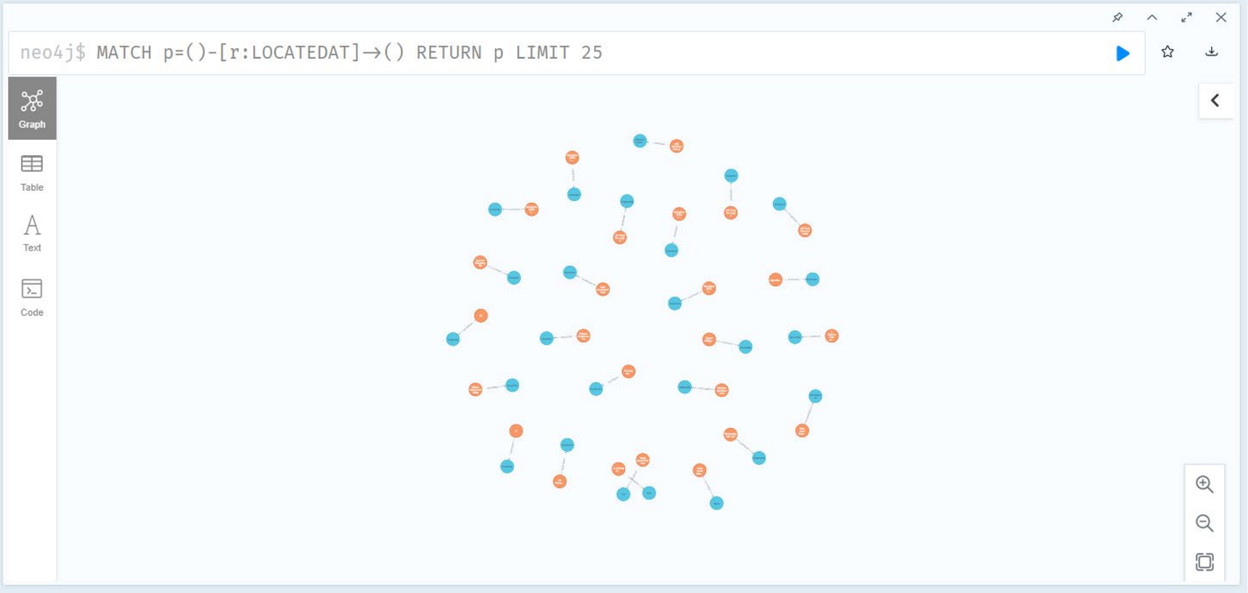
* (49) Listing (25) Reviews (24)

Relationship Types

* (25) HASA (25)

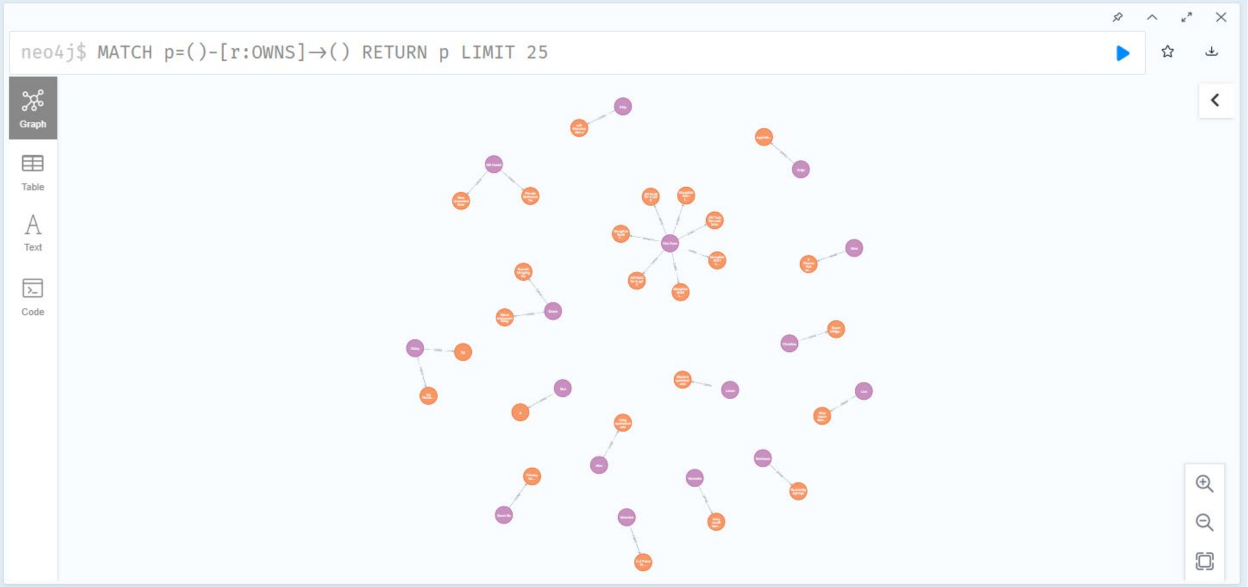
Displaying 49 nodes, 25 relationships.

Relationship between Listing and Address



```
neo4j$ MATCH p=()-[r:LOCATEDAT]→() RETURN p LIMIT 25
```

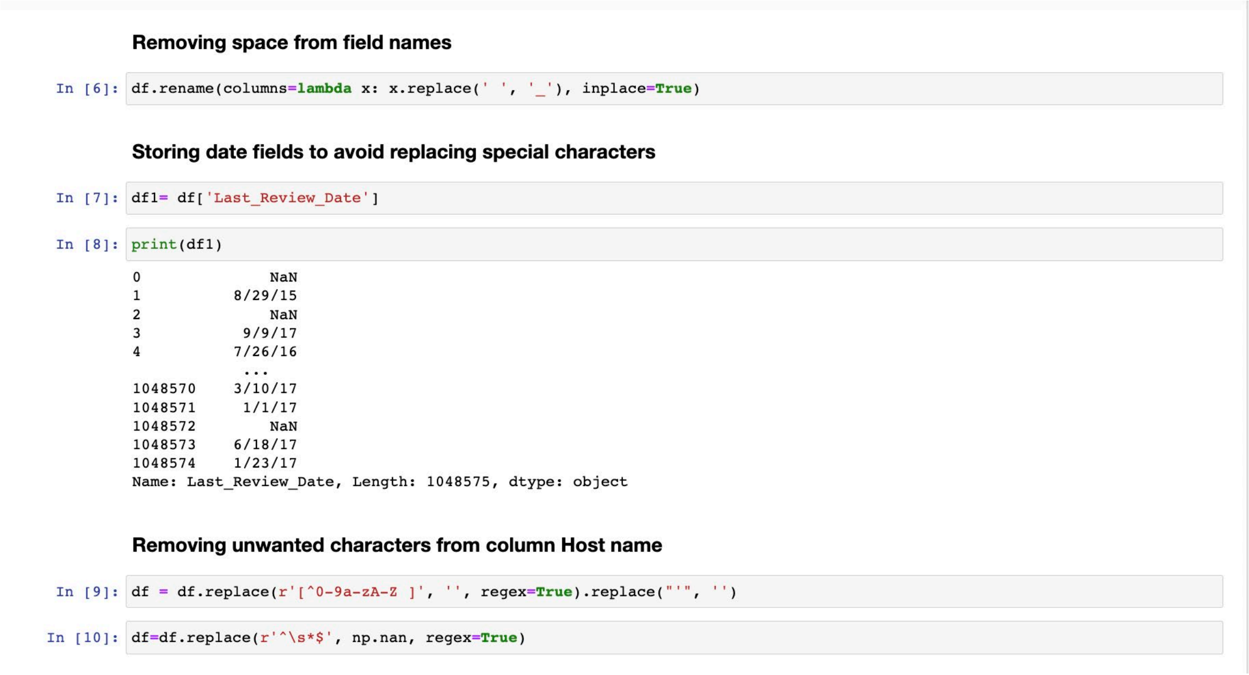
Relationship between Listing and Host



```
neo4j$ MATCH p=()-[r:OWNS]→() RETURN p LIMIT 25
```

3.5. Data Validation and Data Visualization

Validation



Removing space from field names

```
In [6]: df.rename(columns=lambda x: x.replace(' ', '_'), inplace=True)
```

Storing date fields to avoid replacing special characters

```
In [7]: df1= df['Last_Review_Date']
```

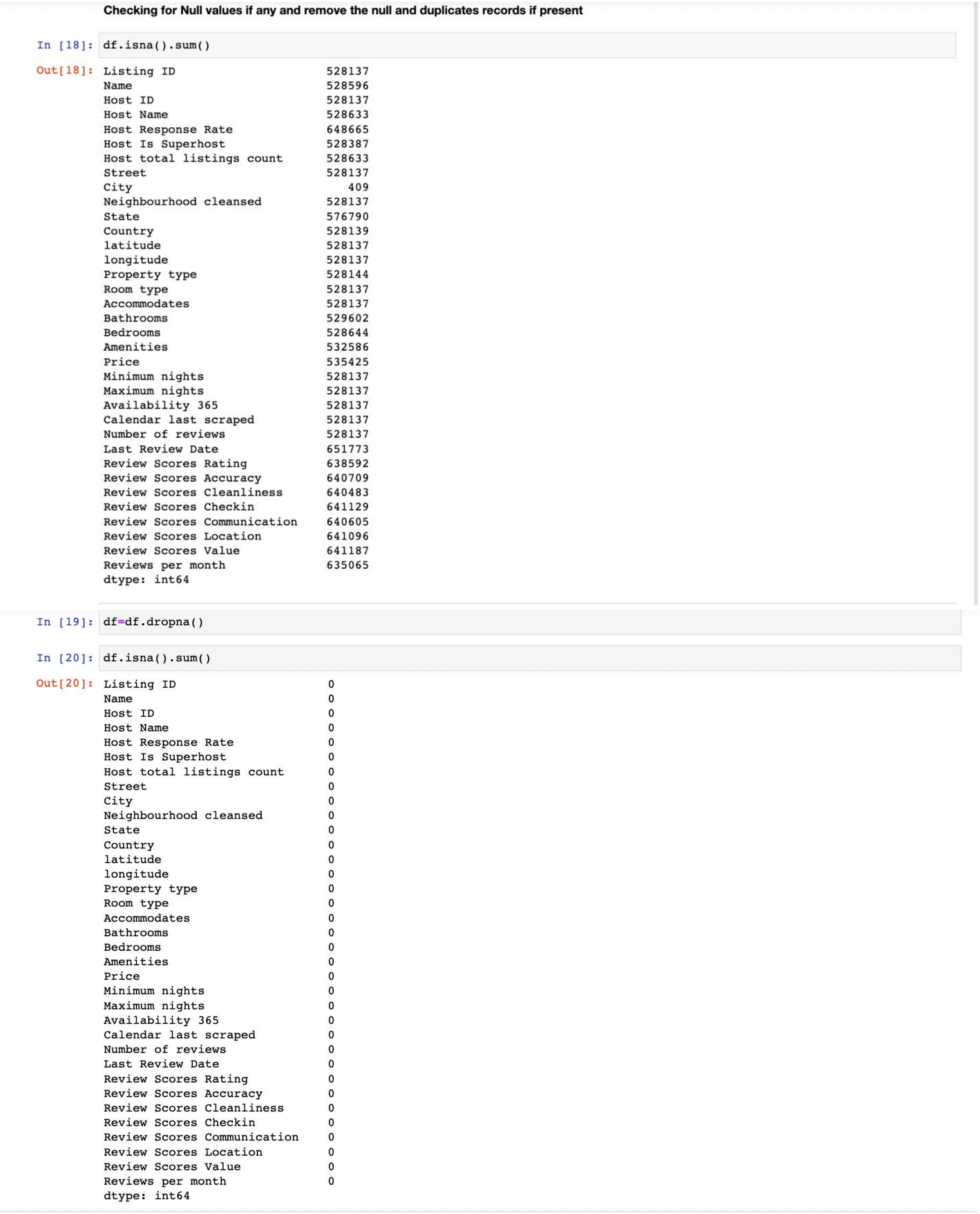
```
In [8]: print(df1)
```

```
0              NaN
1          8/29/15
2              NaN
3           9/9/17
4          7/26/16
            ...
1048570    3/10/17
1048571     1/1/17
1048572        NaN
1048573    6/18/17
1048574    1/23/17
Name: Last_Review_Date, Length: 1048575, dtype: object
```

Removing unwanted characters from column Host name

```
In [9]: df = df.replace(r'[^0-9a-zA-Z ]', '', regex=True).replace("'", '')
```

```
In [10]: df=df.replace(r'^\s*$', np.nan, regex=True)
```

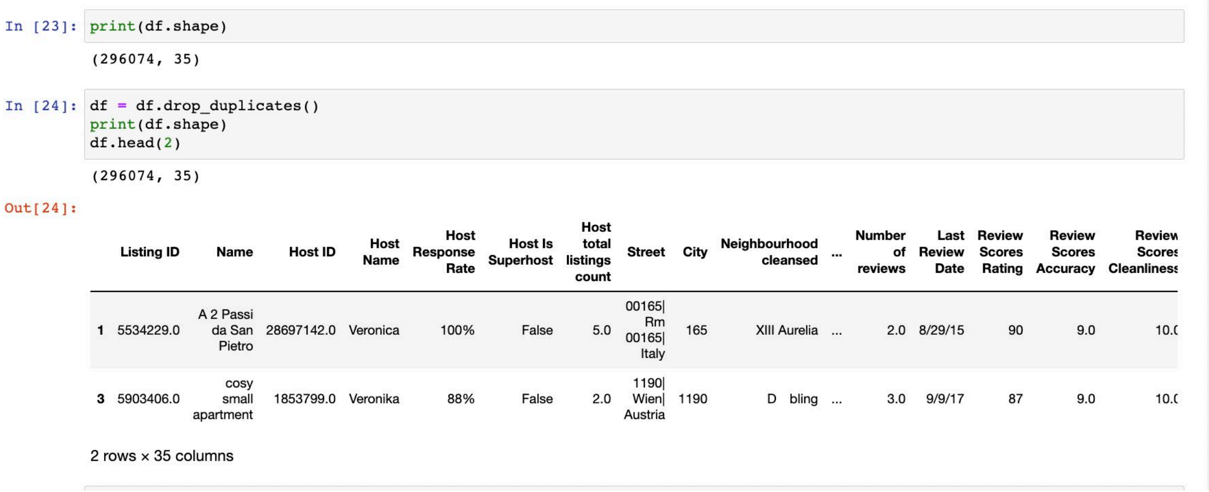


Checking for Null values if any and remove the null and duplicates records if present

```
In [18]: df.isna().sum()

Out[18]: Listing ID                      528137
         Name                            528596
         Host ID                         528137
         Host Name                       528633
         Host Response Rate              648665
         Host Is Superhost               528387
         Host total listings count       528633
         Street                          528137
         City                               409
         Neighbourhood cleansed          528137
         State                           576790
         Country                         528139
         latitude                        528137
         longitude                       528137
         Property type                   528144
         Room type                       528137
         Accommodates                    528137
         Bathrooms                       529602
         Bedrooms                        528644
         Amenities                       532586
         Price                           535425
         Minimum nights                  528137
         Maximum nights                  528137
         Availability 365                528137
         Calendar last scraped           528137
         Number of reviews               528137
         Last Review Date                651773
         Review Scores Rating            638592
         Review Scores Accuracy          640709
         Review Scores Cleanliness       640483
         Review Scores Checkin           641129
         Review Scores Communication     640605
         Review Scores Location          641096
         Review Scores Value             641187
         Reviews per month               635065
         dtype: int64
```

```
In [19]: df=df.dropna()
```

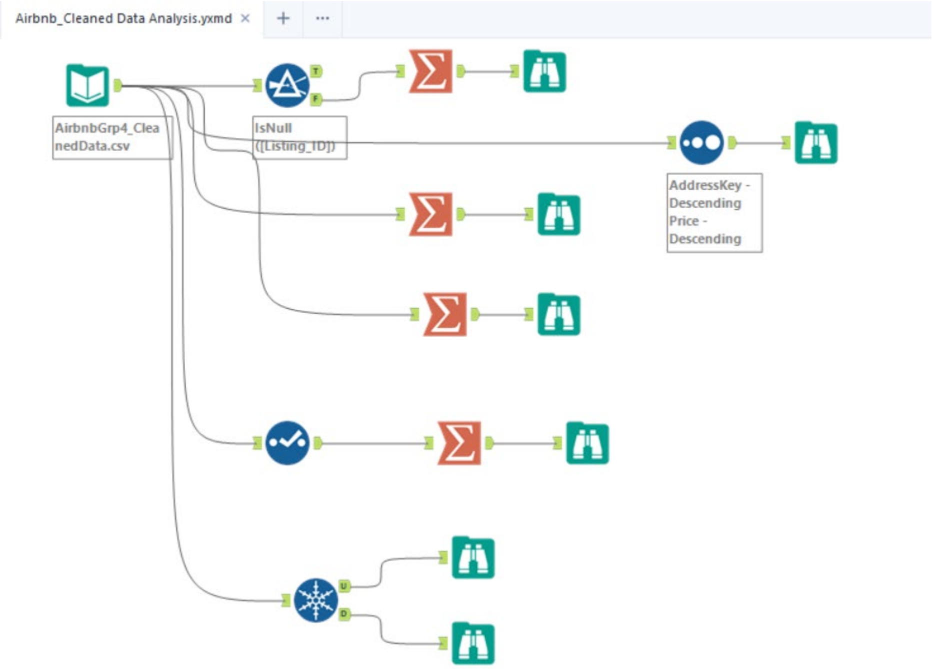
```
In [20]: df.isna().sum()

Out[20]: Listing ID                      0
         Name                            0
         Host ID                         0
         Host Name                       0
         Host Response Rate              0
         Host Is Superhost               0
         Host total listings count       0
         Street                          0
         City                            0
         Neighbourhood cleansed          0
         State                           0
         Country                         0
         latitude                        0
         longitude                       0
         Property type                   0
         Room type                       0
         Accommodates                    0
         Bathrooms                       0
         Bedrooms                        0
         Amenities                       0
         Price                           0
         Minimum nights                  0
         Maximum nights                  0
         Availability 365                0
         Calendar last scraped           0
         Number of reviews               0
         Last Review Date                0
         Review Scores Rating            0
         Review Scores Accuracy          0
         Review Scores Cleanliness       0
         Review Scores Checkin           0
         Review Scores Communication     0
         Review Scores Location          0
         Review Scores Value             0
         Reviews per month               0
         dtype: int64
```



```
In [23]: print(df.shape)

         (296074, 35)

In [24]: df = df.drop_duplicates()
         print(df.shape)
         df.head(2)

         (296074, 35)

Out[24]:
```

	Listing ID	Name	Host ID	Host Name	Host Response Rate	Host Is Superhost	Host total listings count	Street	City	Neighbourhood cleansed	...	Number of reviews	Last Review Date	Review Scores Rating	Review Scores Accuracy	Review Scores Cleanliness
1	5534229.0	A 2 Passi da San Pietro	28697142.0	Veronica	100%	False	5.0	00165| Rm 00165| Italy	165	XIII Aurelia	...	2.0	8/29/15	90	9.0	10.0
3	5903406.0	cosy small apartment	1853799.0	Veronika	88%	False	2.0	1190| Wien| Austria	1190	D bling	...	3.0	9/9/17	87	9.0	10.0

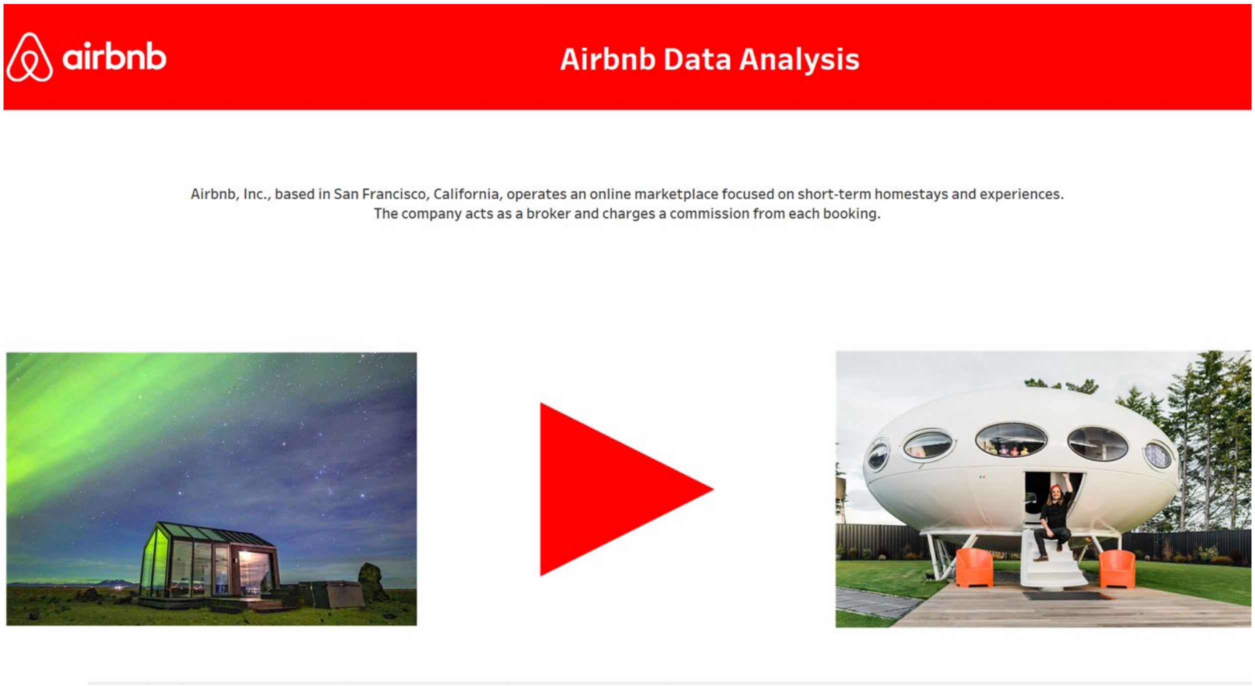
2 rows × 35 columns



Data Profiling done using cleaned to cross validate values populating in the Tableau views Total record count, country wise record counts, null cross-check if generated during the run, summarized the dimensions to check the categorical record count.

Data Visualization

Introduction to Airbnb, Landing Page to the Dashboard

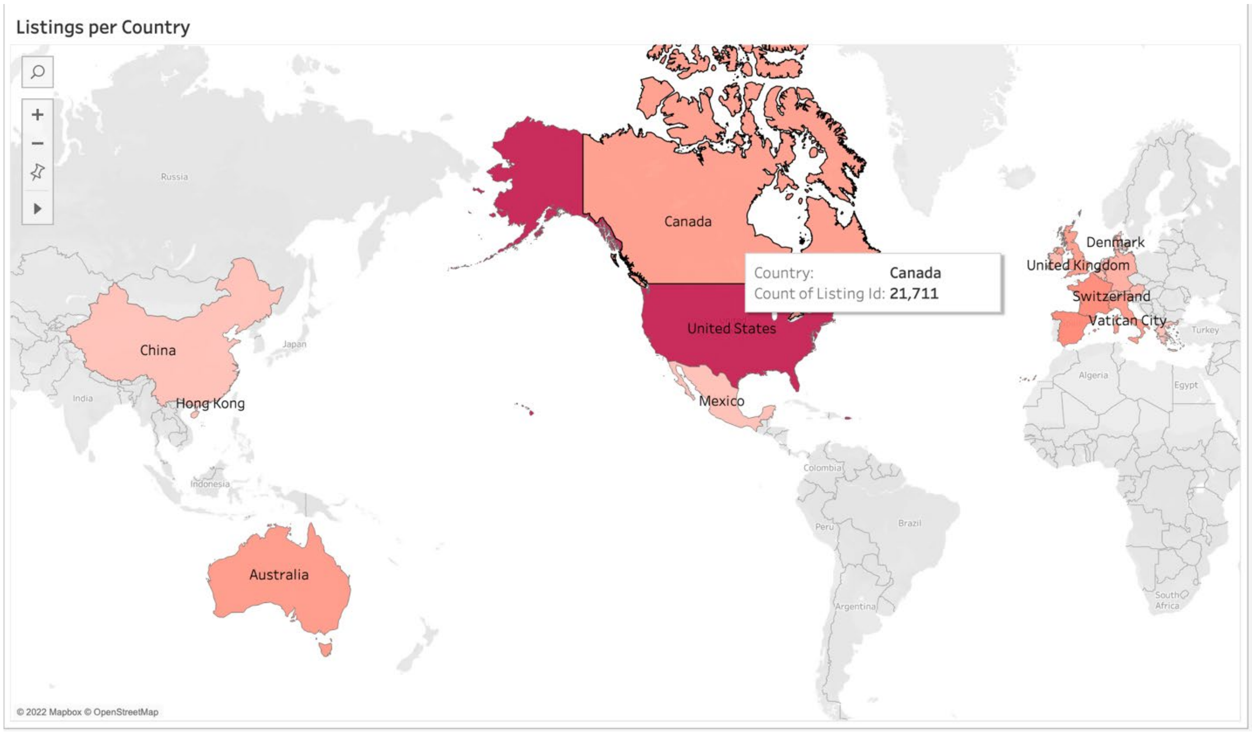


airbnb

Airbnb Data Analysis

Airbnb, Inc., based in San Francisco, California, operates an online marketplace focused on short-term homestays and experiences.
The company acts as a broker and charges a commission from each booking.

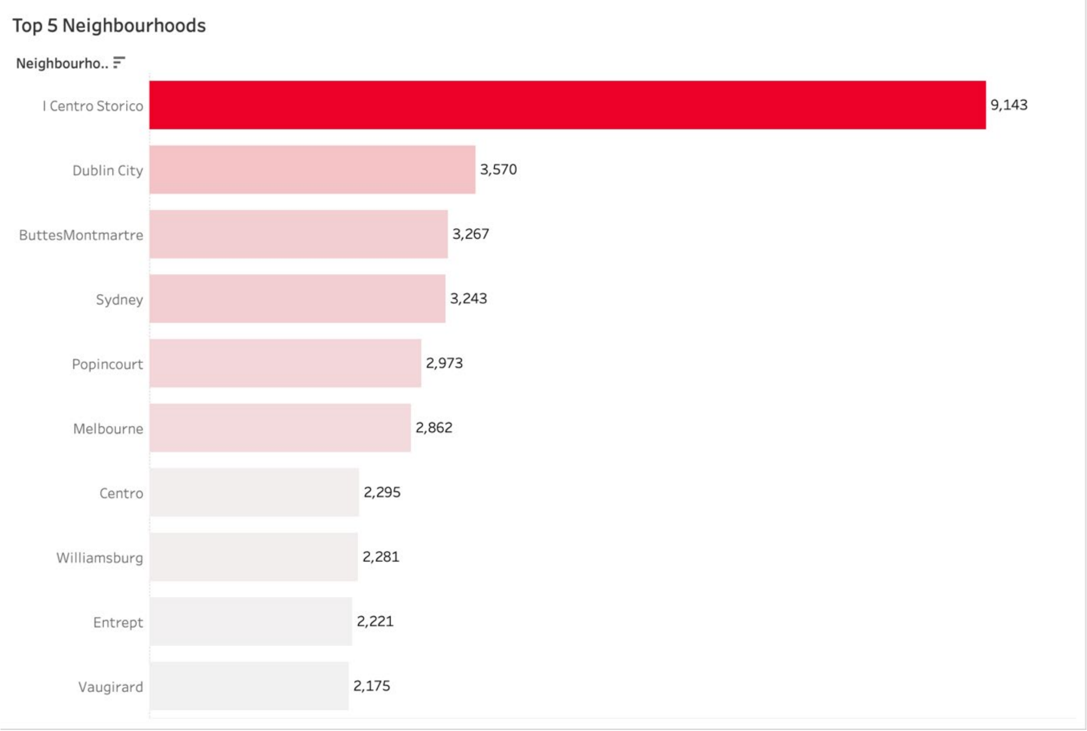
Count of the Listings as per the Country



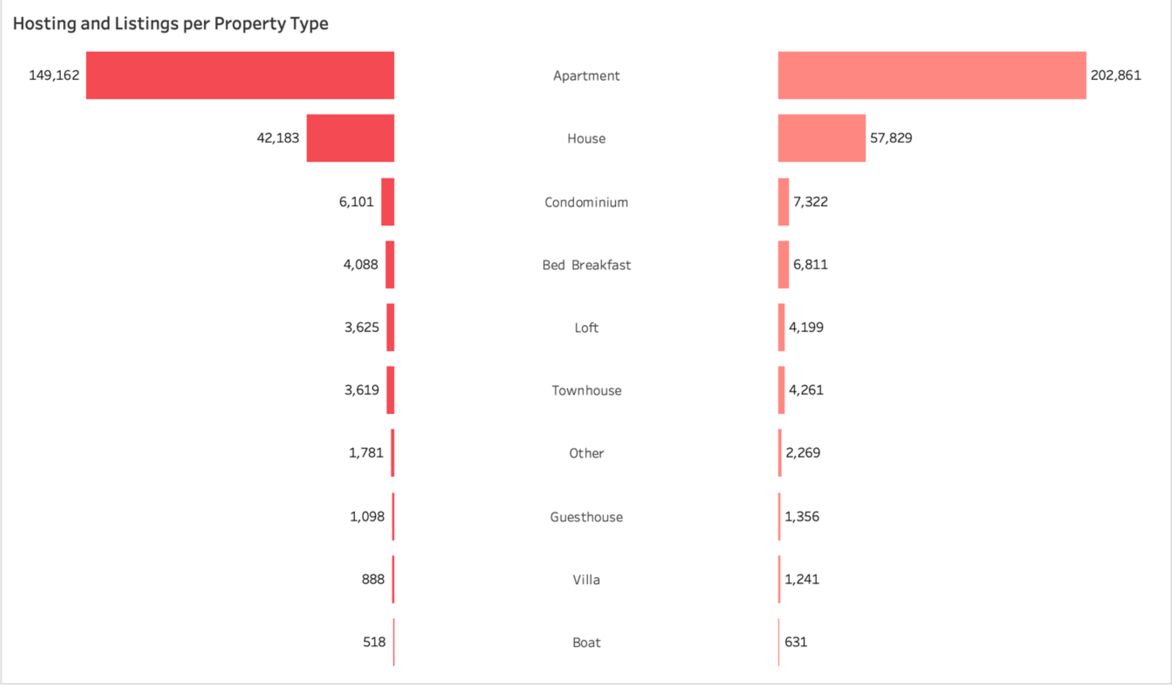
Listings per Country

Country: Canada
Count of Listing Id: 21,711

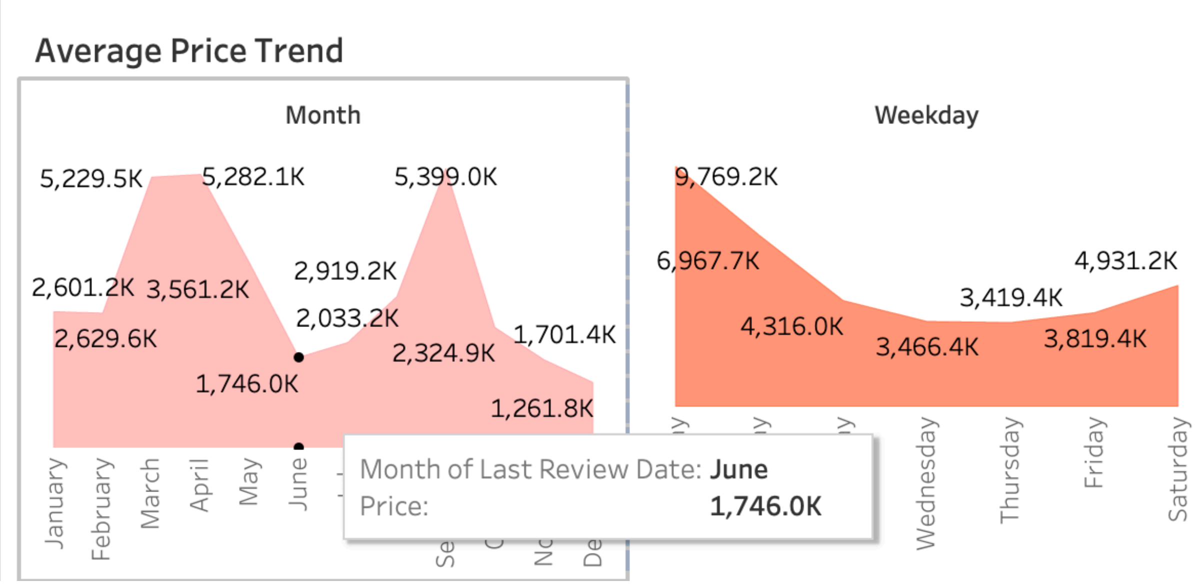
Top 5 Neighbourhoods



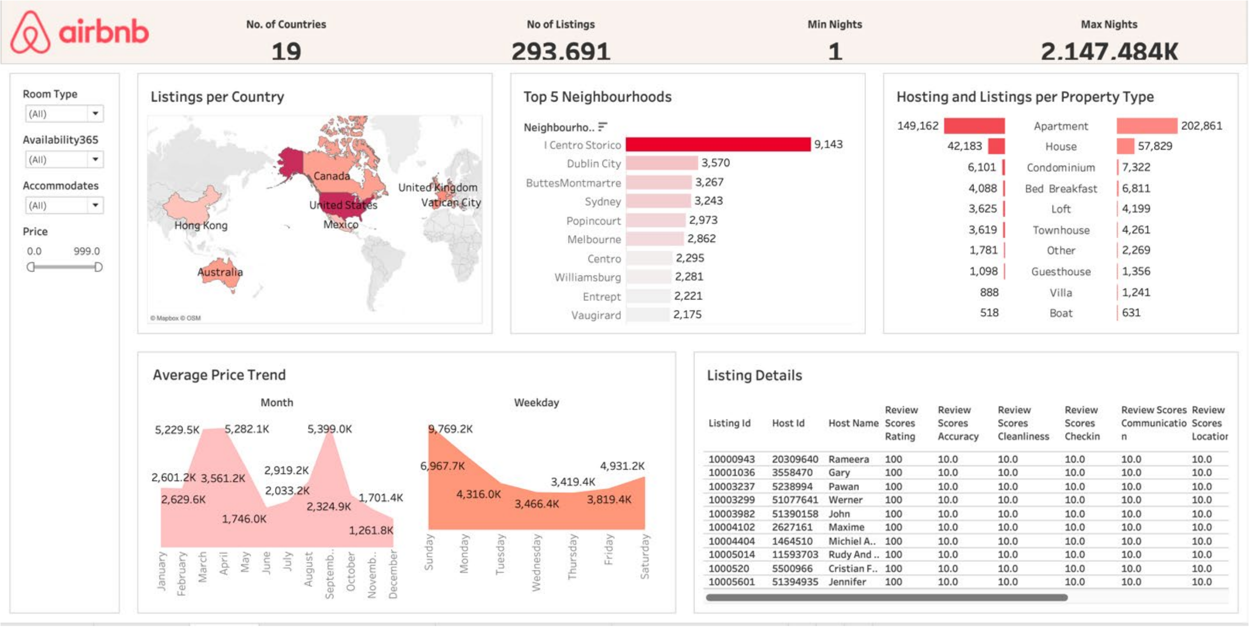
Hosting and Listings Per Property Type



Average Price Trend for a Particular Month and Weekday



Visualizations for Airbnb Dataset



No. of Countries 19

No of Listings 293.691

Min Nights 1

Max Nights 2.147.484K

Room Type (All)

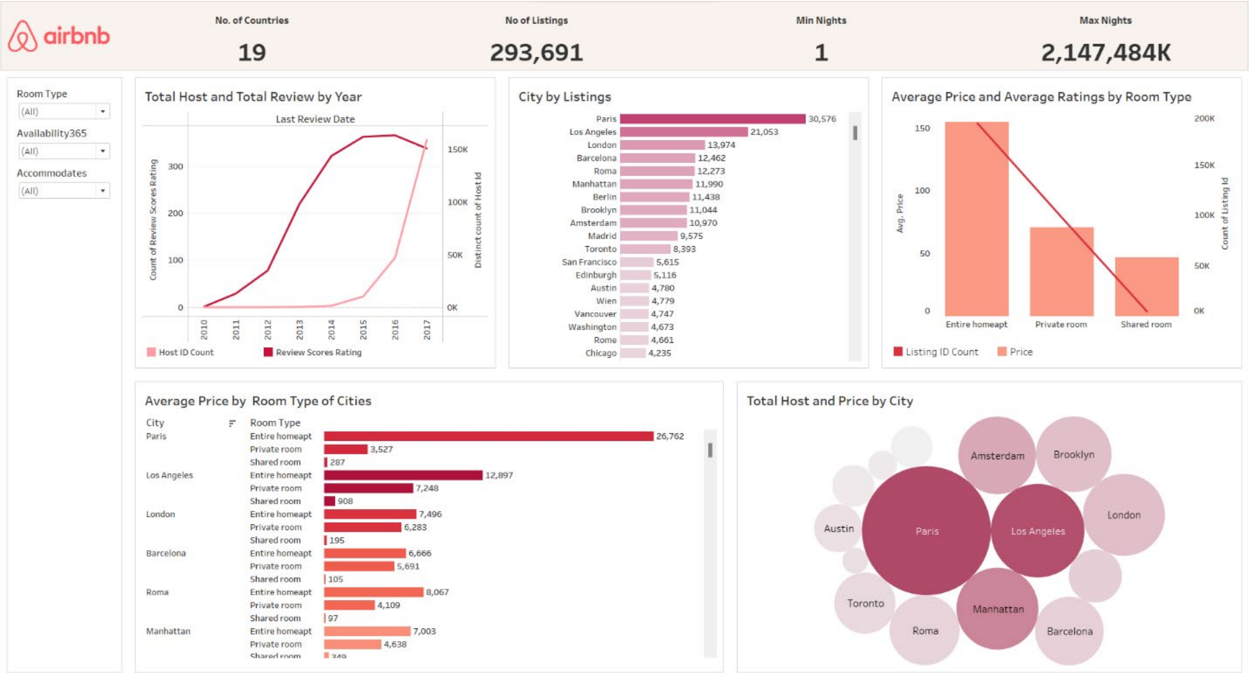
Availability365 (All)

Accommodates (All)

Price 0.0 999.0

Listing Details

Listing Id	Host Id	Host Name	Review Scores Rating	Review Scores Accuracy	Review Scores Cleanliness	Review Scores Checkin	Review Scores Communication	Review Scores Locatior
10000943	20309640	Rameera	100	10.0	10.0	10.0	10.0	10.0
10001036	3558470	Gary	100	10.0	10.0	10.0	10.0	10.0
10003237	5238994	Pawan	100	10.0	10.0	10.0	10.0	10.0
10003299	51077641	Werner	100	10.0	10.0	10.0	10.0	10.0
10003982	51390158	John	100	10.0	10.0	10.0	10.0	10.0
10004102	2627161	Maxime	100	10.0	10.0	10.0	10.0	10.0
10004404	1464510	Michiel A..	100	10.0	10.0	10.0	10.0	10.0
10005014	11593703	Rudy And ..	100	10.0	10.0	10.0	10.0	10.0
1000520	5500966	Cristian F..	100	10.0	10.0	10.0	10.0	10.0
10005601	51394935	Jennifer	100	10.0	10.0	10.0	10.0	10.0



airbnb

No. of Countries: 19

No of Listings: 293,691

Min Nights: 1

Max Nights: 2,147,484K

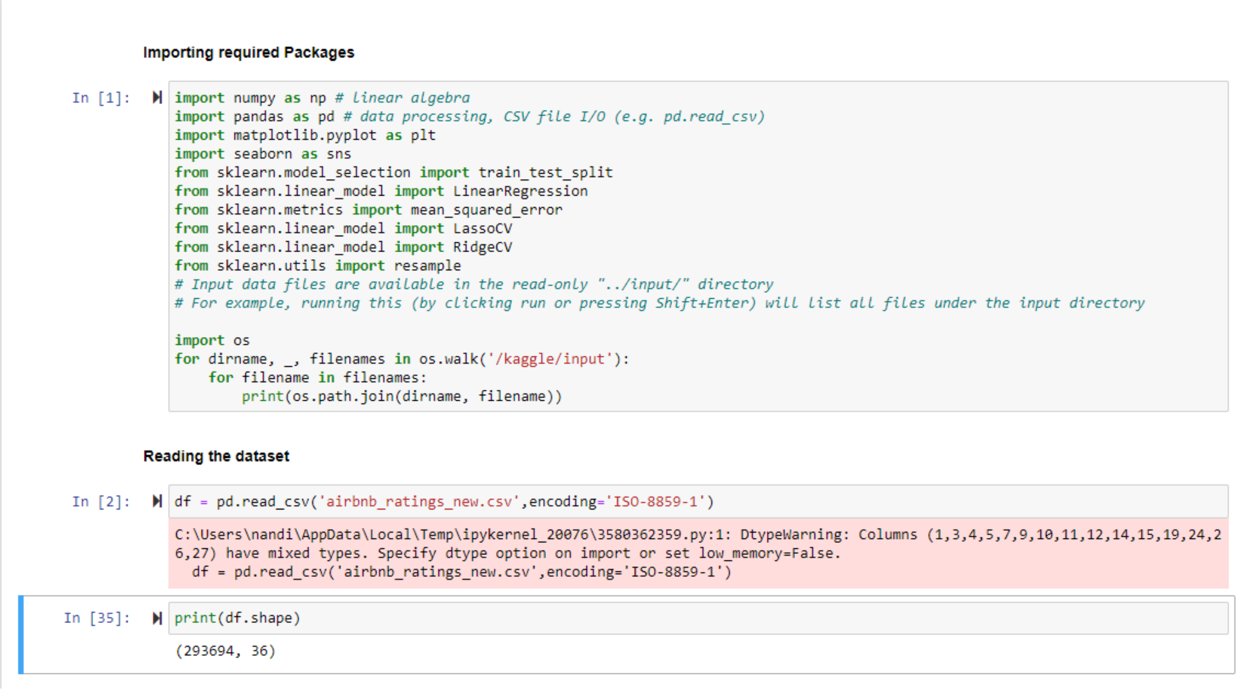
Room Type: (All)

Availability365: (All)

Accommodates: (All)

3.6. System Integration and User Acceptance Testing

Validating the number of records from the Kaggle dataset



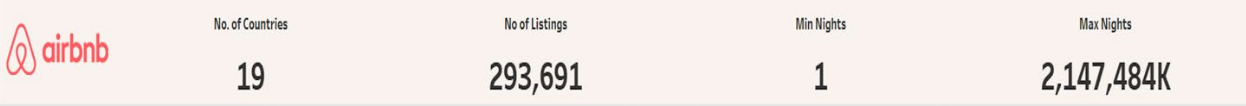
Importing required Packages

```
In [1]:  import numpy as np # linear algebra
         import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
         import matplotlib.pyplot as plt
         import seaborn as sns
         from sklearn.model_selection import train_test_split
         from sklearn.linear_model import LinearRegression
         from sklearn.metrics import mean_squared_error
         from sklearn.linear_model import LassoCV
         from sklearn.linear_model import RidgeCV
         from sklearn.utils import resample
         # Input data files are available in the read-only "../input/" directory
         # For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

         import os
         for dirname, _, filenames in os.walk('/kaggle/input'):
             for filename in filenames:
                 print(os.path.join(dirname, filename))
```

Reading the dataset

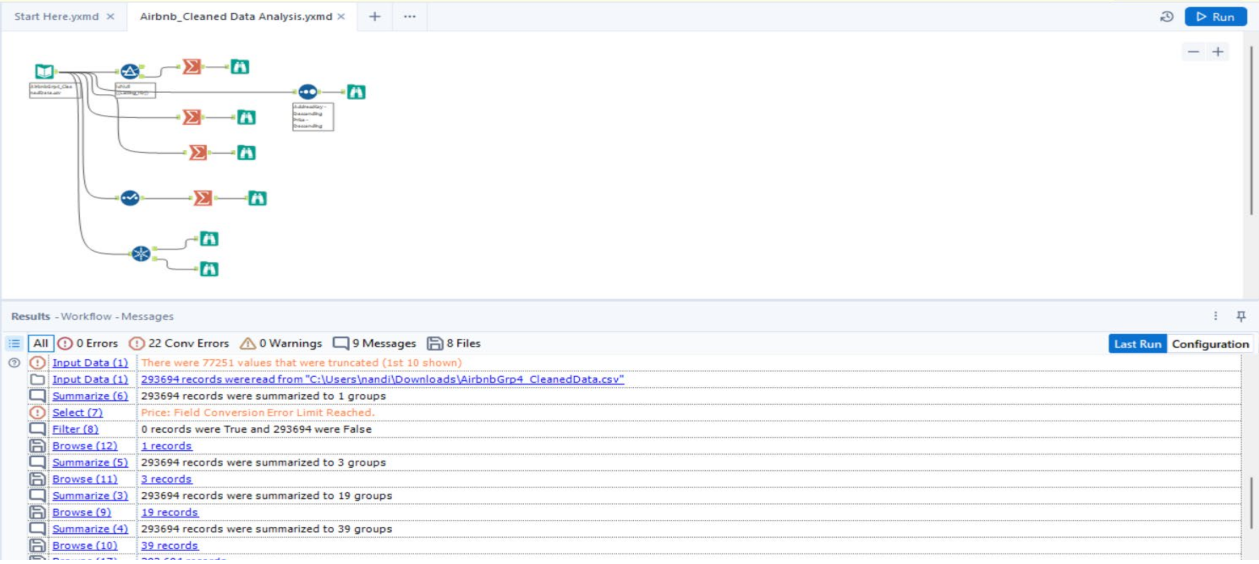
```
In [2]:  df = pd.read_csv('airbnb_ratings_new.csv',encoding='ISO-8859-1')

         C:\Users\nandi\AppData\Local\Temp\ipykernel_20076\3580362359.py:1: DtypeWarning: Columns (1,3,4,5,7,9,10,11,12,14,15,19,24,2
         6,27) have mixed types. Specify dtype option on import or set low_memory=False.
           df = pd.read_csv('airbnb_ratings_new.csv',encoding='ISO-8859-1')

In [35]: print(df.shape)

         (293694, 36)
```

Tableau



	No. of Countries	No of Listings	Min Nights	Max Nights
airbnb	19	293,691	1	2,147,484K

Alteryx



Input Data (1)	There were 77251 values that were truncated (1st 10 shown)
Input Data (1)	293694 records wereread from "C:\Users\nandi\Downloads\AirbnbGrp4_CleanedData.csv"
Summarize (6)	293694 records were summarized to 1 groups
Select (7)	Price: Field Conversion Error Limit Reached.
Filter (8)	0 records were True and 293694 were False
Browse (12)	1 records
Summarize (5)	293694 records were summarized to 3 groups
Browse (11)	3 records
Summarize (3)	293694 records were summarized to 19 groups
Browse (9)	19 records
Summarize (4)	293694 records were summarized to 39 groups
Browse (10)	39 records

3.7. Challenges Encountered

Issue in Loading data in Neo4j
Solution: Increased the heap size

Data Discrepancy while creating nodes/relationship in Neo4j due to spaces in the column
Solution: Created standard naming format for all the columns

Creating keys for weak entities
Solution: Created Surrogate Key

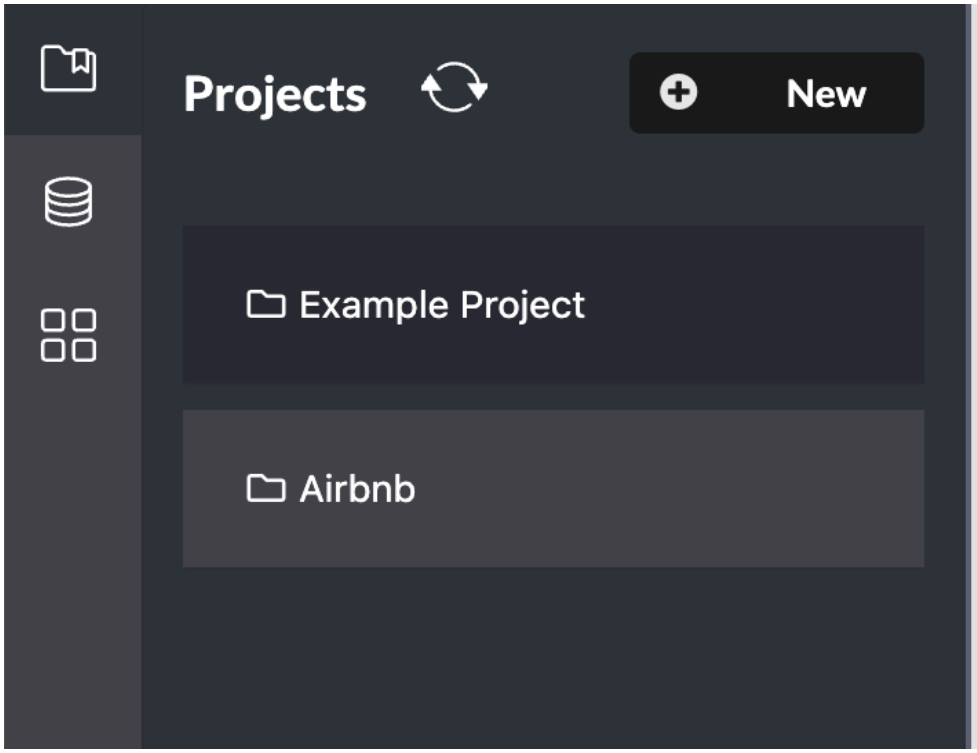
Special characters in the database
Solution: Remove the unwanted characters from the column

Data Load time in Tableau
Solution: Added optional sheet level filters

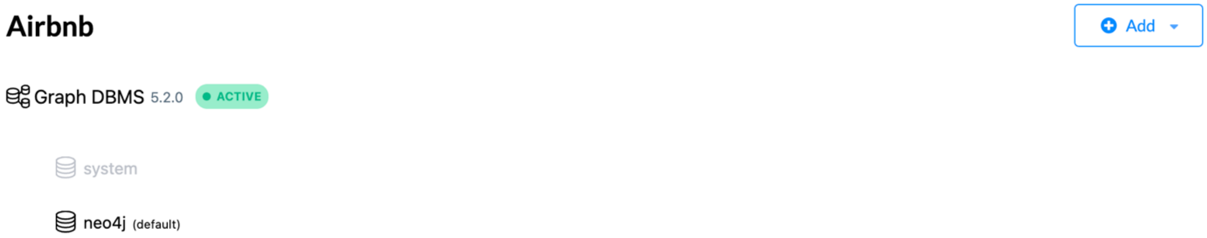
3.8. End User Instructions

Steps for Database Creation and load

1) First, we create a Project from the Neo4j Desktop application.

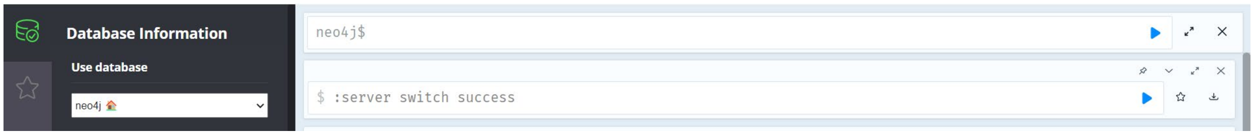


2) Then, we create a local DBMS for our project.



3) Once the DBMS is ready, we can start it by clicking on the start button and then open the server using the open button.

4) After opening the terminal, we can start creating the schema and loading the data into our Neo4j database from the query terminal.

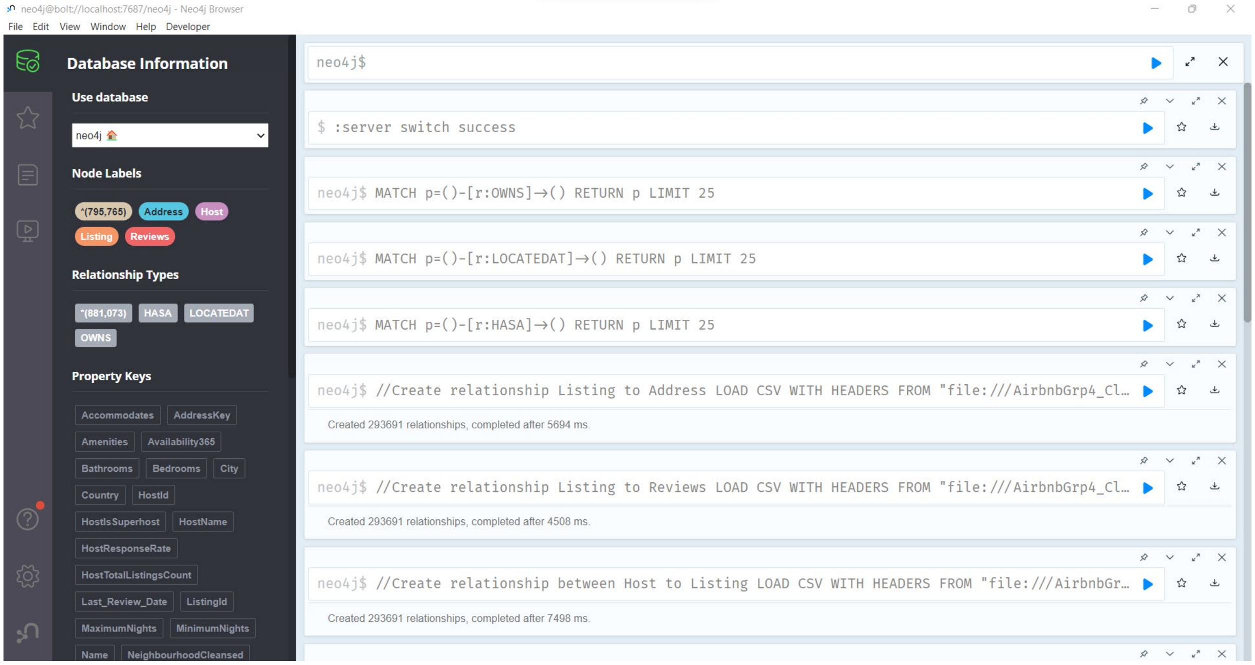


```
neo4j$
$ :server switch success
```

5) For loading the nodes, we need to create all the constraints, it can be created using the code provided in the document.

6) Once we create the constraints for the database, we can start loading the nodes which are unique throughout our dataset

7) Then using the MATCH function, we can load the remaining nodes and their relationship from the CSV.



```
neo4j$
$ :server switch success
neo4j$ MATCH p=()-[r:OWNS]→() RETURN p LIMIT 25
neo4j$ MATCH p=()-[r:LOCATEDAT]→() RETURN p LIMIT 25
neo4j$ MATCH p=()-[r:HASA]→() RETURN p LIMIT 25
neo4j$ //Create relationship Listing to Address LOAD CSV WITH HEADERS FROM "file:///AirbnbGrp4_Cl…
```

Created 293691 relationships, completed after 5694 ms.

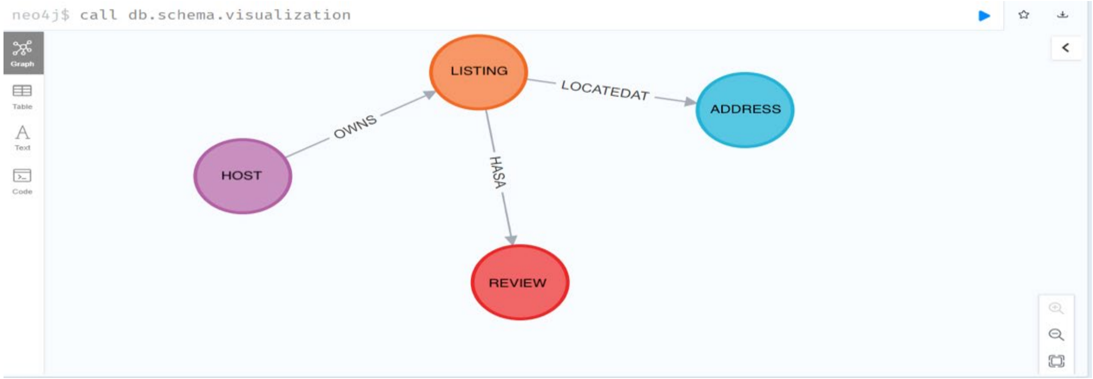
```
neo4j$ //Create relationship Listing to Reviews LOAD CSV WITH HEADERS FROM "file:///AirbnbGrp4_Cl…
```

Created 293691 relationships, completed after 4508 ms.

```
neo4j$ //Create relationship between Host to Listing LOAD CSV WITH HEADERS FROM "file:///AirbnbGr…
```

Created 293691 relationships, completed after 7498 ms.

8) Finally, when all the nodes are created with proper relationship, we can start querying, analyzing, and understanding the data using Graph interface.



```
neo4j$ call db.schema.visualization
```