Airbnb Data Analysis and Big Data Pipeline

OVERVIEW

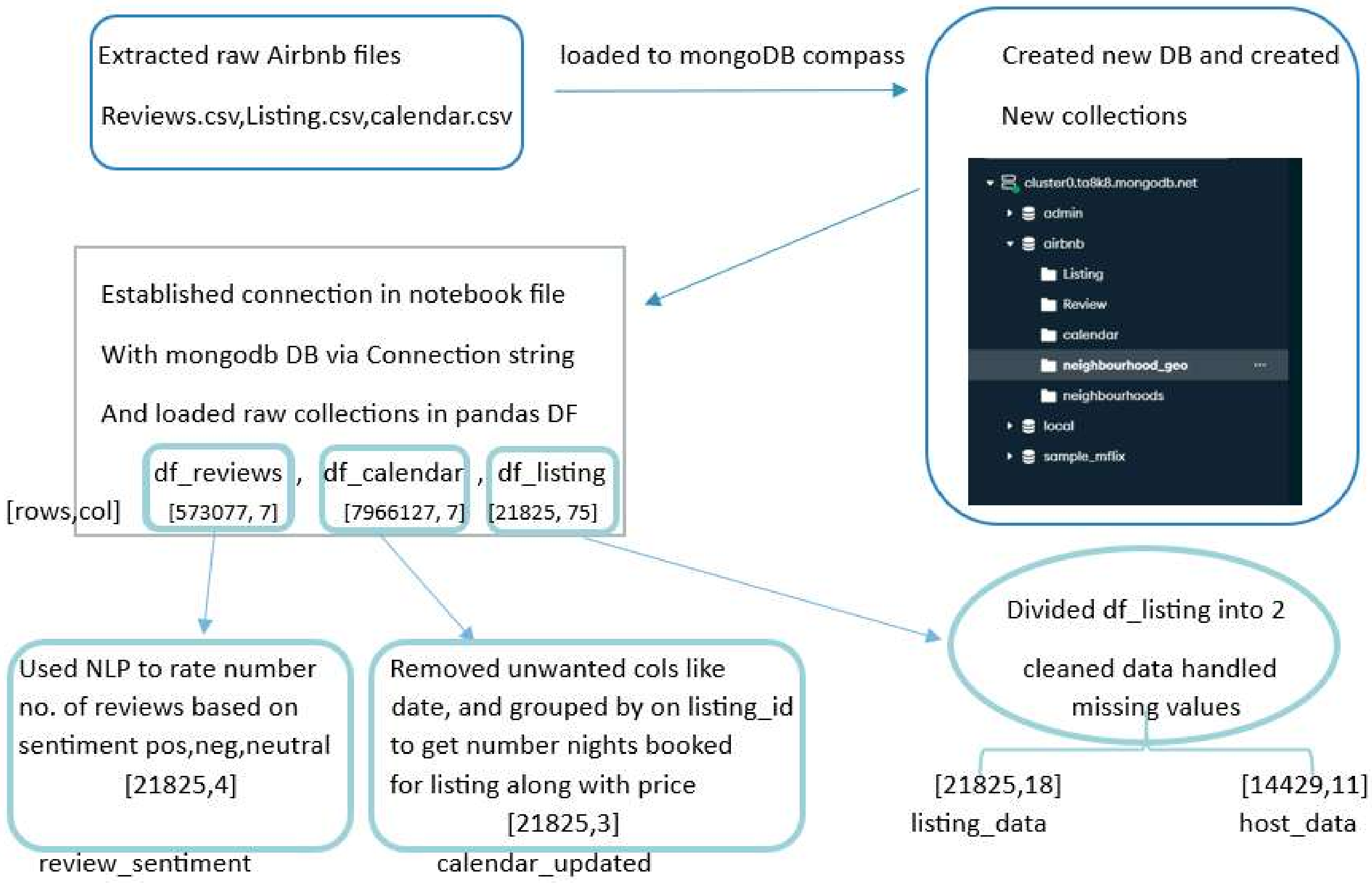
This project centers around the analysis of a large dataset from Airbnb. The dataset provides insights into Airbnb listings in Toronto, including details about the listings, reviews, prices, host information, and availability. The project aims to build a big data pipeline to process, store, and analyze this data using various tools and technologies, such as MongoDB, SQL Server, SSMS, Power BI, and SSIS. The key objective is to perform data extraction, transformation, and loading (ETL), and generate visual insights to support decision-making.

DATASET

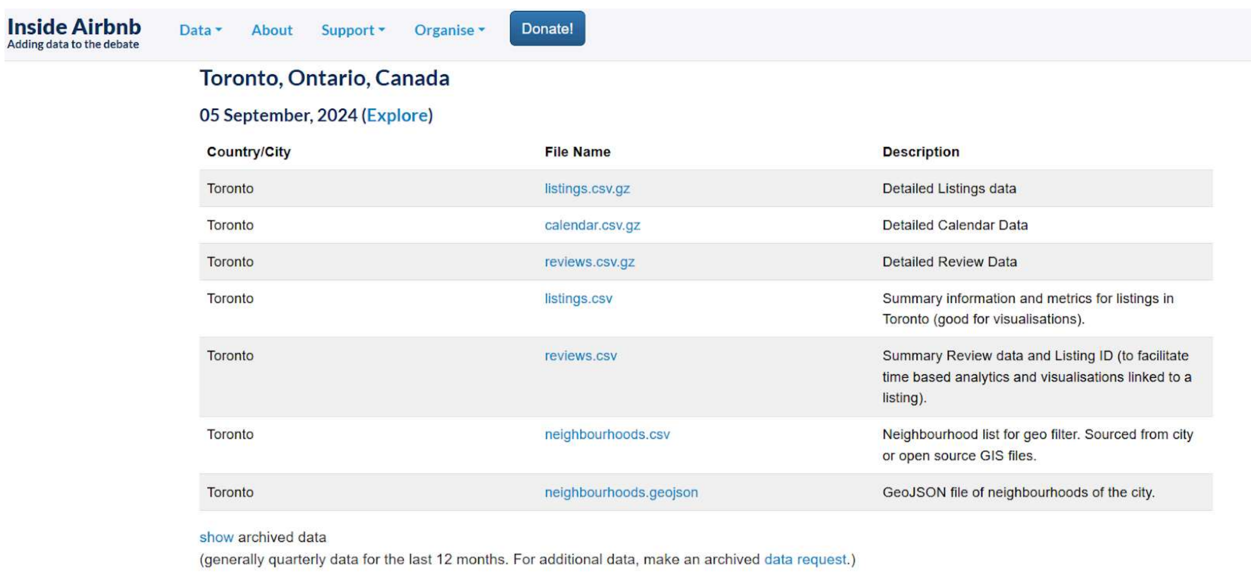
The **Airbnb dataset** sourced from InsideAirbnb includes several key components:

1. **Listings Data**: Contains information about the properties listed on Airbnb (e.g., location, price, amenities, and room types).
2. **Reviews Data**: Includes user comments and ratings for each listing.
3. **Calendar Data**: Provides information on the availability of listings over time (e.g., prices, minimum stay requirements).
4. **Host Data**: Contains details about Airbnb hosts (e.g., host name, response rate, and location).

The project uses these data components to derive insights into customer sentiments, pricing patterns, and host behaviors in the **Toronto** market.



Importing raw tables from source... to mongodb


Inside Airbnb
Adding data to the debate
Data About Support Organise Donate!

Toronto, Ontario, Canada

05 September, 2024 (Explore)

Country/City	File Name	Description
Toronto	listings.csv.gz	Detailed Listings data
Toronto	calendar.csv.gz	Detailed Calendar Data
Toronto	reviews.csv.gz	Detailed Review Data
Toronto	listings.csv	Summary information and metrics for listings in Toronto (good for visualisations).
Toronto	reviews.csv	Summary Review data and Listing ID (to facilitate time based analytics and visualisations linked to a listing).
Toronto	neighbourhoods.csv	Neighbourhood list for geo filter. Sourced from city or open source GIS files.
Toronto	neighbourhoods.geojson	GeoJSON file of neighbourhoods of the city.

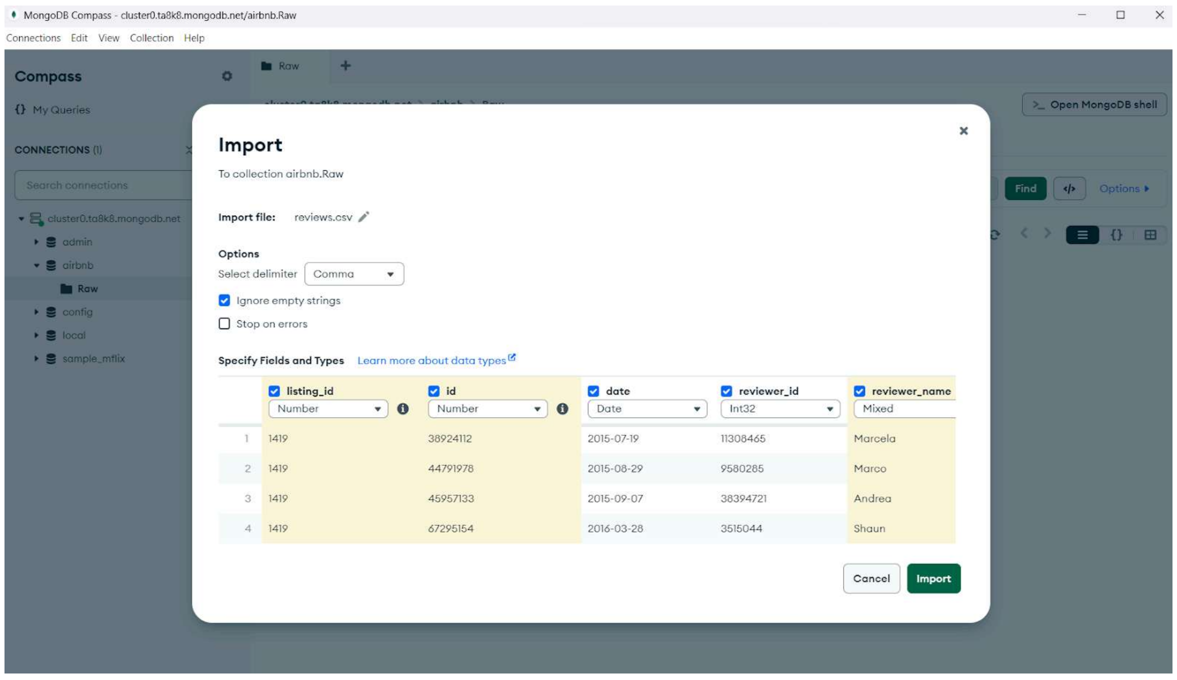
show archived data
(generally quarterly data for the last 12 months. For additional data, make an archived data request.)

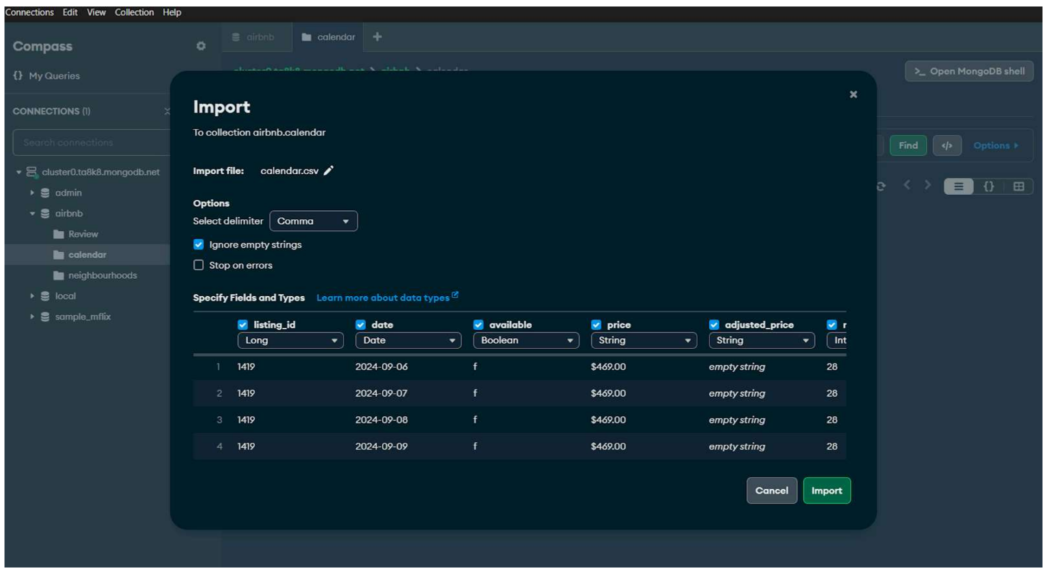
1. **Uploading Data to MongoDB**:

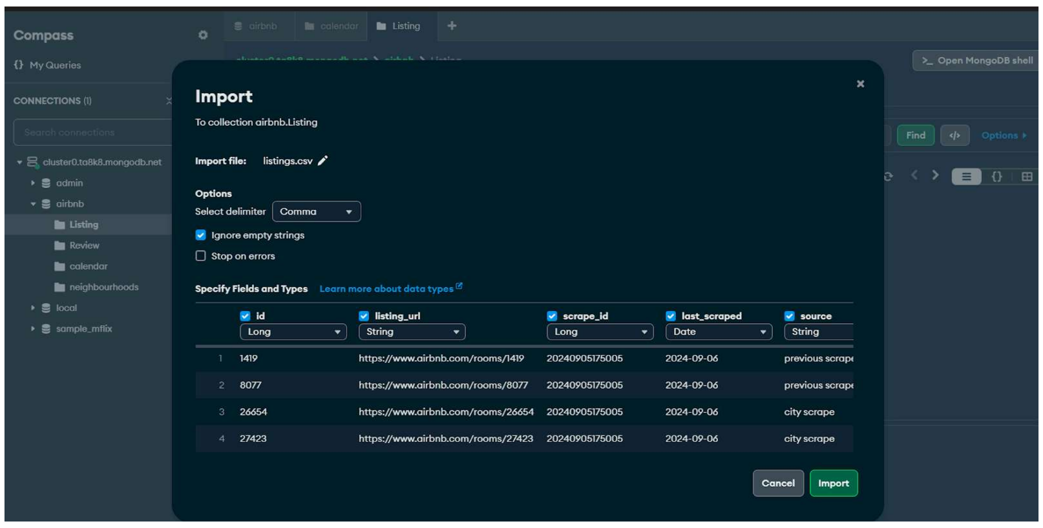
In the initial stage of the project, the raw Airbnb data was loaded into **MongoDB**, a NoSQL database, to take advantage of its flexible, schema-less structure. MongoDB collections were used to store raw data without requiring extensive preprocessing.

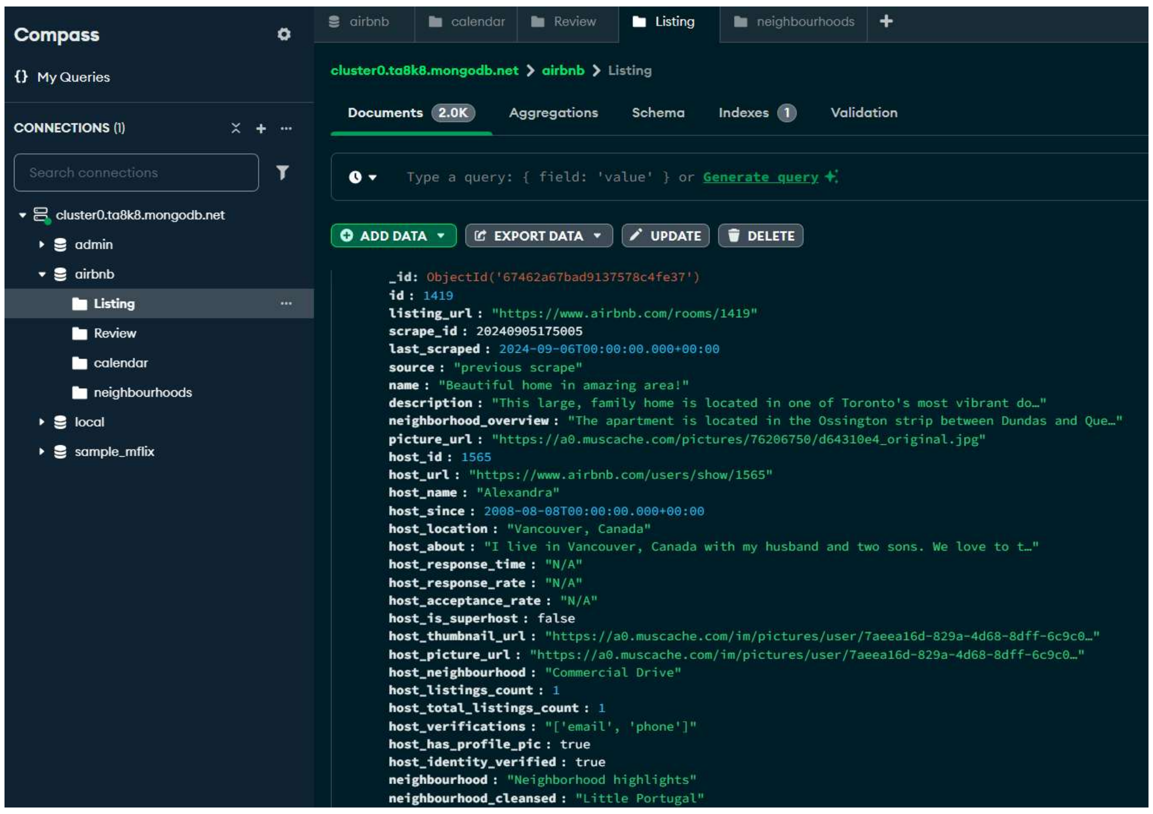
- **Database Name**: airbnb
- **Collections**:
  - listings
  - reviews
  - calendar
  - hosts

This approach allowed easy storage of diverse data types, such as JSON-like documents, and enabled efficient retrieval for analysis.









Connecting to MongoDB from Jupyter Notebook:

After uploading the data to MongoDB, we established a connection using **Python** and the **PyMongo** library. We accessed the MongoDB collections and imported the data into **Pandas DataFrames** for further exploration and preprocessing. This allowed for detailed analysis and manipulation of the raw data.

```
[2] from pymongo import MongoClient
    client=MongoClient('mongodb+srv://azure:1234@cluster0.ta8k8.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0')
    db=client.airbnb
    collection_Review=db.Review
    collection_Listing=db.Listing
    collection_Calendar=db.Calendar
```

```
import pandas as pd

# Fetch all data from the collection
Review = list(collection_Review.find())
Listing = list(collection_Listing.find())
Calendar = list(collection_Calendar.find())

# Create DataFrames
df_Review = pd.DataFrame(Review)
df_Listing = pd.DataFrame(Listing)
df_Calendar = pd.DataFrame(Calendar)
```

df_review

```
[4] df_Review
```

	_id	listing_id	id	date	reviewer_id	reviewer_name	comments
0	67453ac4fdd5b691a09358cc	1419	38924112	2015-07-19	11308465	Marcela	Having the opportunity of arriving to Alexandr...
1	67453ac4fdd5b691a09358cd	1419	44791978	2015-08-29	9580285	Marco	We have no enough words to describe how beauty...
2	67453ac4fdd5b691a09358ce	1419	45957133	2015-09-07	38394721	Andrea	The listing was exceptional and an even better...
3	67453ac4fdd5b691a09358cf	1419	67295154	2016-03-28	3515044	Shaun	Alexandra's home was amazing and in such a nea...
4	67453ac4fdd5b691a09358d0	1419	177702208	2017-08-03	13987100	Kate	Beautiful home. Very comfortable and clean. Pe...
...	...	...	...	...	...	...	...
573072	67453cbbfdd5b691a09c175c	1233097067966383620	1237006458719581379	2024-09-02	131516395	Jairo	Loved this Airbnb, parked the rental in the ga...
573073	67453cbbfdd5b691a09c175d	1233458171350847359	1235643868703935001	2024-08-31	188025909	Jessica Anna	The location is unbeatable, right in the heart...
573074	67453cbbfdd5b691a09c175e	1233634161845475908	1237039990266700869	2024-09-02	293569511	Abishek	Great Place! Very clean and comfortable. I wil...
573075	67453cbbfdd5b691a09c175f	1233751078363326332	1239177218317027784	2024-09-05	582901404	Omar	Thank you for hosting my family. we had a gre...
573076	67453cbbfdd5b691a09c1760	1234042847646185352	1238562455560871223	2024-09-04	497990924	Hyungjun	Clean and comfy room and very kind host, I hig...

573077 rows × 7 columns

```
df_Review.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 573077 entries, 0 to 573076
Data columns (total 7 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   _id            573077 non-null  object
 1   listing_id     573077 non-null  int64
 2   id             573077 non-null  int64
 3   date           573077 non-null  datetime64[ns]
 4   reviewer_id    573077 non-null  int64
 5   reviewer_name  573077 non-null  object
 6   comments       573077 non-null  object
dtypes: datetime64[ns](1), int64(3), object(3)
memory usage: 30.6+ MB
```

- ➔ Created new df out of df_review with new sentiment column

```python
from textblob import TextBlob
import pandas as pd


def analyze_sentiment(comment):
    comment = str(comment)
    if not comment or pd.isnull(comment):  # Handle blank or NaN comments
        return "neutral"
    analysis = TextBlob(comment)
    if analysis.sentiment.polarity > 0:
        return "positive"
    elif analysis.sentiment.polarity < 0:
        return "negative"
    else:
        return "neutral"

df_sentiment = df_Review.drop(columns=['_id', 'id', 'date', 'reviewer_name'])
df_sentiment['sentiment'] = df_sentiment['comments'].apply(analyze_sentiment)
```

```python
[15] df_sentiment['sentiment'].value_counts()
```

sentiment	count
positive	521105
neutral	45781
negative	6191

dtype: int64

We used the TextBlob library for its simplicity and effectiveness in performing sentiment analysis on user comments. By dropping unnecessary columns ('_id', 'id', 'date', 'reviewer_name') from our DataFrame, we focused on the relevant data. We then analyzed the sentiment of the 'comments' column and added a new 'sentiment' column to the DataFrame, indicating whether each comment was positive, negative, or neutral.

-> Next, we analyzed the sentiment of reviews for each listing by grouping the data by listing_id and counting the occurrences of positive, negative, and neutral sentiments. Using pandas' groupby and unstack functions, we created a new DataFrame, Review_sentiment, which summarizes the sentiment distribution for each listing. This structured data will aid in identifying trends in customer feedback and inform our report. We also exported the results to a CSV file.

```python
Review_sentiment = df_sentiment.groupby(['listing_id', 'sentiment']).size().unstack(fill_value=0)

Review_sentiment.reset_index(inplace=True)

Review_sentiment.columns.name = None
Review_sentiment.rename(columns={'positive': 'positive', 'negative': 'negative', 'neutral': 'neutral'}, inplace=True)

Review_sentiment
```

	listing_id	negative	neutral	positive
0	1419	0	0	6
1	8077	0	1	168
2	26654	1	1	40
3	27423	0	1	29
4	30931	0	0	1
...	...	...	...	...
16605	1233097067966383620	0	0	1
16606	1233458171350847359	0	0	1
16607	1233634161845475908	0	0	1
16608	1233751078363326332	0	0	1
16609	1234042847646185352	0	0	1

16610 rows × 4 columns

Next steps: Generate code with Review_sentiment | View recommended plots | New interactive sheet

```python
[20] Review_sentiment.to_csv('review_sentiment.csv', index=False)
```

Now onto next part which is df_Calender total rows almost 8 million, we only pick insights what we want

df_Calendar DataFrame containing approximately 8 million rows and 7 columns. It includes data on listing IDs, dates, availability, prices, and minimum night required to stay by host. The data types indicate a mix of integers, floats, and objects, reflecting the diverse nature of the dataset.

```
[23] df_Calendar
```

	listing_id	date	available	price	adjusted_price	minimum_nights	maximum_nights
0	1419	2024-09-06	f	$469.00	NaN	28.0	730.0
1	1419	2024-09-07	f	$469.00	NaN	28.0	730.0
2	1419	2024-09-08	f	$469.00	NaN	28.0	730.0
3	1419	2024-09-09	f	$469.00	NaN	28.0	730.0
4	1419	2024-09-10	f	$469.00	NaN	28.0	730.0
...	...	...	...	...	...	...	...
7966122	1238622881503213641	2025-09-01	f	$425.00	NaN	28.0	365.0
7966123	1238622881503213641	2025-09-02	f	$425.00	NaN	28.0	365.0
7966124	1238622881503213641	2025-09-03	f	$425.00	NaN	28.0	365.0
7966125	1238622881503213641	2025-09-04	f	$425.00	NaN	28.0	365.0
7966126	1238622881503213641	2025-09-05	f	$425.00	NaN	28.0	365.0

7966127 rows × 7 columns

```
df_Calendar.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7966127 entries, 0 to 7966126
Data columns (total 7 columns):
 #   Column          Dtype
---  ------          -----
 0   listing_id      int64
 1   date            object
 2   available       object
 3   price           object
 4   adjusted_price  object
 5   minimum_nights  float64
 6   maximum_nights  float64
dtypes: float64(2), int64(1), object(4)
memory usage: 425.4+ MB
```

On looking the data calendar has info about each listing day wise for past year 24/23 we will only pick price per listing and number of days it was available in last calendar year

Also we found out that for two listing price changes thus we take average of the price as per frequency

```
#distinct listing_id and price
df_Calendar[['listing_id','price']].value_counts()
```

listing_id	price	count
984169057959518938	$88.00	366
1214184360723826950	$59.00	366
936094881610665141	$450.00	365
935643228222733696	$699.00	365
933529625658376226	$81.00	365
...	...	...
39173764	$1,200.00	89
	$950.00	25
963208358673073891	$899.00	25
39173764	$1,100.00	21
	$1,000.00	14

21830 rows × 1 columns

dtype: int64

```
[35] price_counts = df_Calendar.groupby('listing_id')['price'].nunique()
     listing_ids_with_multiple_prices = price_counts[price_counts > 1].index.tolist()
     print(listing_ids_with_multiple_prices)
```

```
[39173764, 963208358673073891]
```

We transformed the original df_Calendar DataFrame by grouping it by listing_id and price, and aggregated the count of 'f' in the available column to create a new column called nights_booked. Also saved the result to a CSV file named calender_updated.csv.

```
[43] calendar_update = df_Calendar.groupby(['listing_id', 'price']).agg(nights_booked=('available', lambda x: (x == 'f').sum())).reset_index()
```

```
[44] calendar_update
```

	listing_id	price	nights_booked
0	1419	$469.00	365
1	8077	$75.00	365
2	26654	$155.00	291
3	27423	$75.00	365
4	30931	$100.00	365
...	...	...	...
21820	1238375993354181658	$350.00	0
21821	1238381731476165839	$89.00	23
21822	1238429354932685108	$170.00	311
21823	1238458106144360234	$150.00	319
21824	1238622881503213641	$425.00	76

21825 rows × 3 columns

Next steps: Generate code with calendar_result | View recommended plots | New interactive sheet

```
#save calender_result into a csv named calender_updated
calendar_update.to_csv('calender_updated.csv',index=False)
```

Now onto df_listing

```
df_Listing.head()
```

	id	listing_url	scrape_id	last_scraped	source	name	description	neighborhood_overview	picture_url	host_id	...	review_scores_communication	review_scores_location	review_score
0	1419	https://www.airbnb.com/rooms/1419	20240905175005	2024-09-06	previous scrape	Beautiful home in amazing area!	This large, family home is located in one of T...	The apartment is located in the Ossington stri...	https://a0.muscache.com/pictures/76206750/d643...	1565	...	5.00	5.00	
1	8077	https://www.airbnb.com/rooms/8077	20240905175005	2024-09-06	previous scrape	Downtown Harbourfront Private Room	Guest room in a luxury condo with access to al...	NaN	https://a0.muscache.com/pictures/11780344/141c...	22795	...	4.90	4.92	
2	26654	https://www.airbnb.com/rooms/26654	20240905175005	2024-09-06	city scrape	World Class @ CN Tower, convention centre, The...	CN Tower, TIFF Bell Lightbox, Metro Convention...	There's a reason they call it the Entertainmen...	https://a0.muscache.com/pictures/81811785/5dcd...	113345	...	4.76	4.86	
3	27423	https://www.airbnb.com/rooms/27423	20240905175005	2024-09-06	city scrape	Executive Studio Unit- Ideal for One Person	Brand new, fully furnished studio basement apa...	NaN	https://a0.muscache.com/pictures/176936/b687ed...	118124	...	5.00	4.87	
4	30931	https://www.airbnb.com/rooms/30931	20240905175005	2024-09-06	previous scrape	Downtown Toronto - Waterview Condo	Split level waterfront condo with a breathtaki...	NaN	https://a0.muscache.com/pictures/227971/e8ebd7...	22795	...	NaN	NaN	

5 rows × 75 columns

```
df_Listing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21825 entries, 0 to 21824
Data columns (total 75 columns):
 #   Column                                        Non-Null Count  Dtype
---  ------                                        --------------  -----
 0   id                                            21825 non-null  int64
 1   listing_url                                   21825 non-null  object
 2   scrape_id                                     21825 non-null  int64
 3   last_scraped                                  21825 non-null  object
 4   source                                        21825 non-null  object
 5   name                                          21825 non-null  object
 6   description                                   21316 non-null  object
 7   neighborhood_overview                         10759 non-null  object
 8   picture_url                                   21825 non-null  object
 9   host_id                                       21825 non-null  int64
 10  host_url                                      21825 non-null  object
 11  host_name                                     21823 non-null  object
 12  host_since                                    21823 non-null  object
 13  host_location                                 16235 non-null  object
 14  host_about                                    10114 non-null  object
 15  host_response_time                            15741 non-null  object
 16  host_response_rate                            15741 non-null  object
 17  host_acceptance_rate                          16297 non-null  object
 18  host_is_superhost                             20914 non-null  object
 19  host_thumbnail_url                            21823 non-null  object
 20  host_picture_url                              21823 non-null  object
 21  host_neighbourhood                            8221 non-null   object
 22  host_listings_count                           21823 non-null  float64
 23  host_total_listings_count                     21823 non-null  float64
 24  host_verifications                            21823 non-null  object
 25  host_has_profile_pic                          21823 non-null  object
 26  host_identity_verified                        21823 non-null  object
 27  neighbourhood                                 10760 non-null  object
 28  neighbourhood_cleansed                        21825 non-null  object
 29  neighbourhood_group_cleansed                  0 non-null      float64
 30  latitude                                      21825 non-null  float64
 31  longitude                                     21825 non-null  float64
 32  property_type                                 21825 non-null  object
 33  room_type                                     21825 non-null  object
 34  accommodates                                  21825 non-null  int64
 35  bathrooms                                     16524 non-null  float64
 36  bathrooms_text                                21810 non-null  object
 37  bedrooms                                      20185 non-null  float64
 38  beds                                          16519 non-null  float64
 39  amenities                                     21825 non-null  object
 40  price                                         16536 non-null  object
 41  minimum_nights                                21825 non-null  int64
 42  maximum_nights                                21825 non-null  int64
 43  minimum_minimum_nights                        21825 non-null  int64
 44  maximum_minimum_nights                        21825 non-null  int64
 45  minimum_maximum_nights                        21825 non-null  int64
 46  maximum_maximum_nights                        21825 non-null  int64
 47  minimum_nights_avg_ntm                        21825 non-null  float64
 48  maximum_nights_avg_ntm                        21825 non-null  float64
 49  calendar_updated                              0 non-null      float64
 50  has_availability                              20774 non-null  object
 51  availability_30                               21825 non-null  int64
 52  availability_60                               21825 non-null  int64
```

It provides info on **Airbnb listings** in Toronto, detailing both **host** and **listing** attributes.

For better analysis and organization, we propose splitting the data into two tables:

1. **host_data**: This table will focus on host-specific information, including host_id, host_name, host_since, and host_response_time,etc
2. **listings_data**: This table will contain details about the listings, such as id, location, rooms, availability, and review data,etc

By dividing the data in this manner, we can easily explore host behavior and listing trends, while removing unnecessary columns (like URLs,timestamps,etc) to simplify the dataset for more focused analysis.

```
[133] listing_data=df_Listing[['id','name','host_id','neighbourhood_cleansed','latitude','longitude','property_type','room_type','accommodates','bathrooms','bathrooms_text','bedrooms','

[134] host_data=df_Listing[['host_id','host_name','host_since','host_location','host_response_time','host_response_rate','host_acceptance_rate','host_is_superhost','host_listings_count'

[135] host_data=host_data.drop_duplicates()

[136] host_data.info()

<class 'pandas.core.frame.DataFrame'>
Index: 14429 entries, 0 to 21820
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   host_id               14429 non-null  int64
 1   host_name             14427 non-null  object
 2   host_since            14427 non-null  object
 3   host_location         10610 non-null  object
 4   host_response_time    9215 non-null   object
 5   host_response_rate    9215 non-null   object
 6   host_acceptance_rate  9612 non-null   object
 7   host_is_superhost     14034 non-null  object
 8   host_listings_count   14427 non-null  float64
 9   host_verifications    14427 non-null  object
 10  host_identity_verified  14427 non-null  object
dtypes: float64(1), int64(1), object(9)
memory usage: 1.3+ MB

listing_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21825 entries, 0 to 21824
Data columns (total 18 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   id                      21825 non-null  int64
 1   name                    21825 non-null  object
 2   host_id                 21825 non-null  int64
 3   neighbourhood_cleansed  21825 non-null  object
 4   latitude                21825 non-null  float64
 5   longitude               21825 non-null  float64
 6   property_type           21825 non-null  object
 7   room_type               21825 non-null  object
 8   accommodates            21825 non-null  int64
 9   bathrooms               16524 non-null  float64
 10  bathrooms_text          21810 non-null  object
 11  bedrooms                20185 non-null  float64
 12  beds                    16519 non-null  float64
 13  amenities               21825 non-null  object
 14  minimum_nights          21825 non-null  int64
 15  has_availability        20774 non-null  object
 16  license                 10730 non-null  object
 17  instant_bookable        21825 non-null  object
dtypes: float64(5), int64(4), object(9)
memory usage: 3.0+ MB
```

Handling missing values for column license,bathrooms, bedrooms,beds,has_availability

NaN values in license & has_availibility were replaced by ‘unlicensed’ and ‘f’ respectively

To handle missing values in ,bathrooms, bedrooms,beds more complex ML based approach was used

```python
[130] # Create a new column 'bathrooms' by extracting the numeric part from 'bathrooms_text'
      df_Listing['bathrooms'] = df_Listing['bathrooms_text'].str.extract(r'(\d+\.?\d*)')[0].astype(float)

      # Handle specific cases for 'Private half-bath' and 'Shared half-bath'
      df_Listing.loc[df_Listing['bathrooms_text'].str.contains('Private half-bath', na=False), 'bathrooms'] = 0.5
      df_Listing.loc[df_Listing['bathrooms_text'].str.contains('Shared half-bath', na=False), 'bathrooms'] = 0.5
```

```python
[138] def estimate_bedrooms(row):
          if pd.isna(row['bedrooms']):
              if row['room_type'] == 'Entire home/apt':
                  return max(1, row['accommodates'] // 2)
              elif row['room_type'] == 'Private room':
                  return 1
              elif row['room_type'] == 'Shared room':
                  return 1
              elif row['room_type'] == 'Hotel room':
                  return 1
          return row['bedrooms']

      listing_data['bedrooms'] = listing_data.apply(estimate_bedrooms, axis=1)
```

```python
[139] #save host_data and listing_data in csv
      host_data.to_csv('host_data.csv',index=False)
      listing_data.to_csv('listing_data.csv',index=False)
```

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
```

```python
df_non_null = df[df['bedrooms'].notnull()]
df_null = df[df['bedrooms'].isnull()]

X = df_non_null[['accommodates', 'bathrooms', 'beds']]
y = df_non_null['bedrooms']

X['beds'].fillna(X['beds'].median(), inplace=True)
df_null['beds'].fillna(df_null['beds'].median(), inplace=True)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print(f"RMSE: {np.sqrt(mean_squared_error(y_test, y_pred))}")

X_null = df_null[['accommodates', 'bathrooms', 'beds']]
df_null['bedrooms'] = model.predict(X_null).round().astype(int)

df.loc[df['bedrooms'].isnull(), 'bedrooms'] = df_null['bedrooms']

print(df[['accommodates', 'bathrooms', 'beds', 'bedrooms']].head(10))
```

```python
df_non_null_beds = df[df['beds'].notnull()]
df_null_beds = df[df['beds'].isnull()]

X_beds = df_non_null_beds[['property_type', 'room_type', 'accommodates', 'bathrooms', 'bedrooms']]
y_beds = df_non_null_beds['beds']

X_beds = pd.get_dummies(X_beds, drop_first=True)
df_null_beds = pd.get_dummies(df_null_beds, drop_first=True)

X_beds, df_null_beds = X_beds.align(df_null_beds, join='left', axis=1, fill_value=0)

X_train_beds, X_test_beds, y_train_beds, y_test_beds = train_test_split(X_beds, y_beds, test_size=0.2, random_state=42)

model_beds = RandomForestRegressor(n_estimators=100, random_state=42)
model_beds.fit(X_train_beds, y_train_beds)

y_pred_beds = model_beds.predict(X_test_beds)
print(f"RMSE: {np.sqrt(mean_squared_error(y_test_beds, y_pred_beds))}")

X_null_beds = df_null_beds[X_beds.columns]
df_null_beds['beds'] = model_beds.predict(X_null_beds).round().astype(int)

df.loc[df['beds'].isnull(), 'beds'] = df_null_beds['beds']
```
✓ 2.1s

```python
host_data.info()
```
✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14429 entries, 0 to 14428
Data columns (total 11 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   host_id                 14429 non-null  int64
 1   host_name               14427 non-null  object
 2   host_since              14427 non-null  object
 3   host_location           10610 non-null  object
 4   host_response_time      9215 non-null   object
 5   host_response_rate      9215 non-null   object
 6   host_acceptance_rate    9612 non-null   object
 7   host_is_superhost       14034 non-null  object
 8   host_listings_count     14427 non-null  float64
 9   host_verifications      14427 non-null  object
 10  host_identity_verified  14427 non-null  object
dtypes: float64(1), int64(1), object(9)
memory usage: 1.2+ MB
```

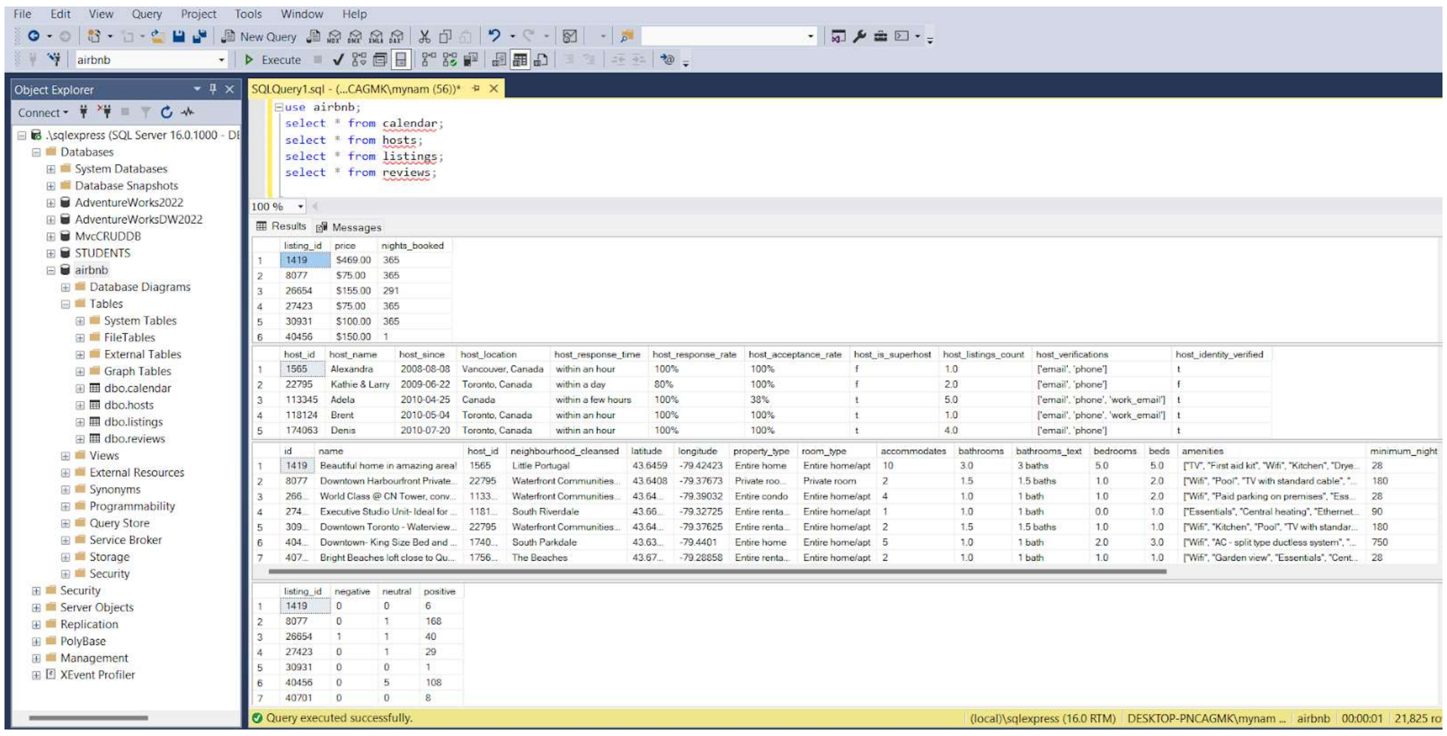
Missing values in host_data were handled for the columns host_location, host_response_time, host_response_rate, and host_acceptance_rate by employing appropriate imputation techniques such as filling with placeholders, averages, medians, or the most frequent values, depending on the context and data type.

Part -2 (SSMS) query + combining tables + analysing + exploration

After the data extraction and transformation process, we uploaded the clean data to a database schema (AirbnbDB) in an SQL server through an automated process in Python. An extract of the program is shown below.

```python
import pyodbc
import pandas as pd
import os

# SQL Server connection configuration
server = r'DESKTOP-IG574ON\SQLEXPRESS'  # Change this to your server name
database = 'AirbnbDB'  # Database name

# Create the database if it does not exist
connection_string_master = f"DRIVER={{ODBC Driver 17 for SQL Server}};SERVER={server};DATABASE=master;Trusted_Connection=yes;"
conn_master = pyodbc.connect(connection_string_master)
conn_master.autocommit = True  # Enable autocommit for CREATE DATABASE
cursor_master = conn_master.cursor()

# Create the database
print(f"Creating database '{database}' if it does not exist...")
cursor_master.execute(f"IF NOT EXISTS (SELECT name FROM sys.databases WHERE name = '{database}') CREATE DATABASE {database}")
cursor_master.close()
conn_master.close()

# Connect to the new database
connection_string = f"DRIVER={{ODBC Driver 17 for SQL Server}};SERVER={server};DATABASE={database};Trusted_Connection=yes;"
conn = pyodbc.connect(connection_string)
cursor = conn.cursor()

# SQL query to create tables
table_creation_query = """
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[reviews]') AND type in (N'U'))
BEGIN
    CREATE TABLE reviews (
        listing_id BIGINT,
        negative INT,
        neutral INT,
```

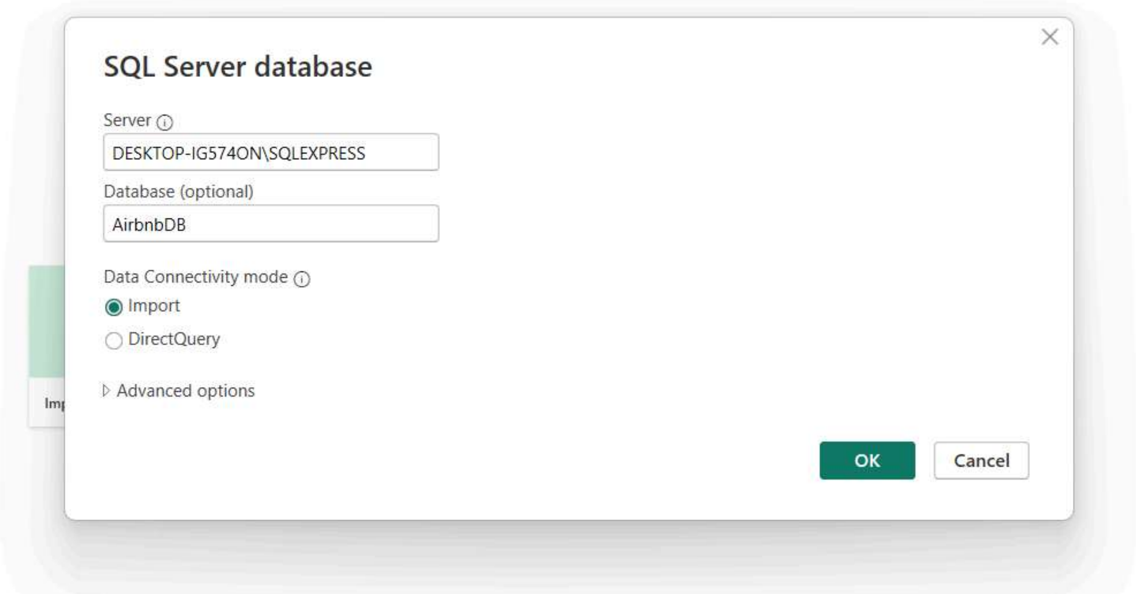
After saving respective dataframes to csv locally. We moved on to SSMS in order to create a new Database "airbnb" and load these csv as tables under schema for further analysis and exploration :



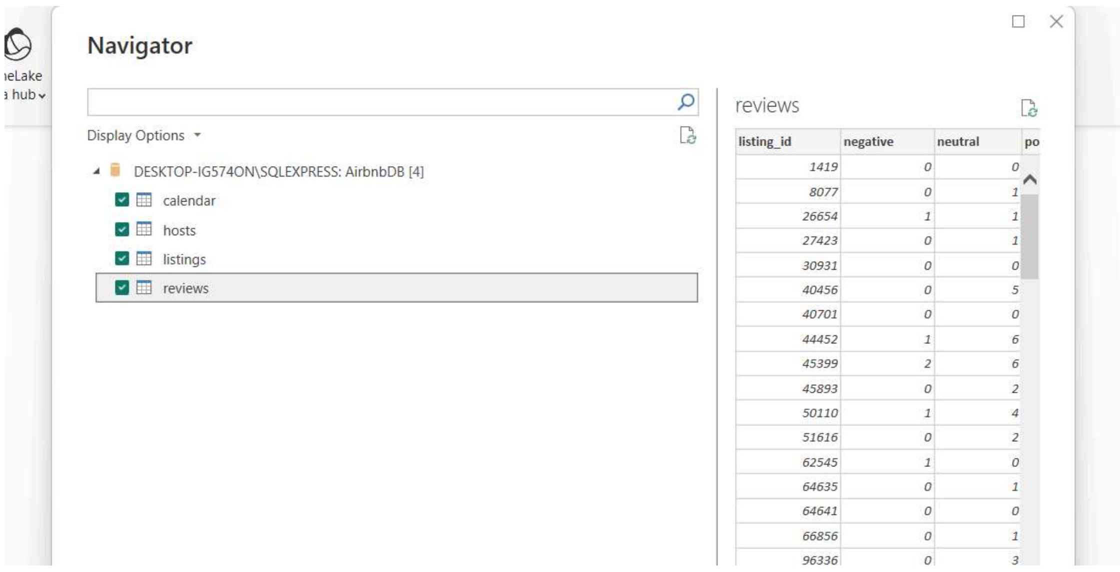
Part -3 SQL Server Connection and Visualization

Through ETL processes, we obtain the necessary information to create various visualizations aimed at transforming data into valuable insights for the business. These visualizations allow us to identify key patterns and trends clearly and effectively, supporting informed decision-making. For this process, we use Microsoft Power BI, which provides advanced capabilities for data exploration and presentation.

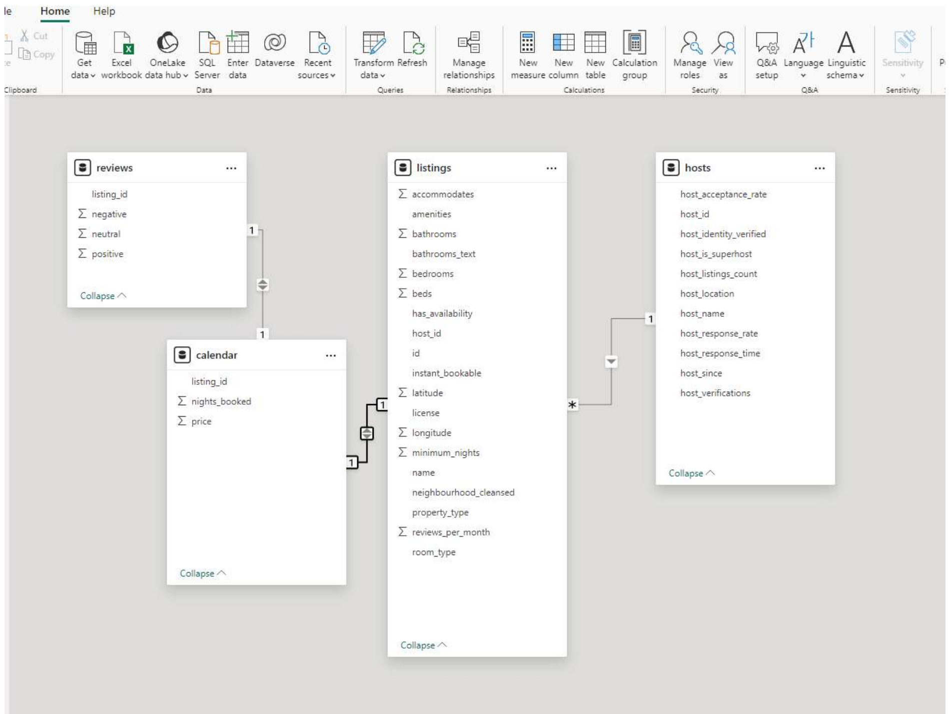
To proceed with creating the visualizations, it is necessary to configure the connection to the data source. In this case, from Microsoft Power BI, we configure a connection to our SQL server to access the required information.



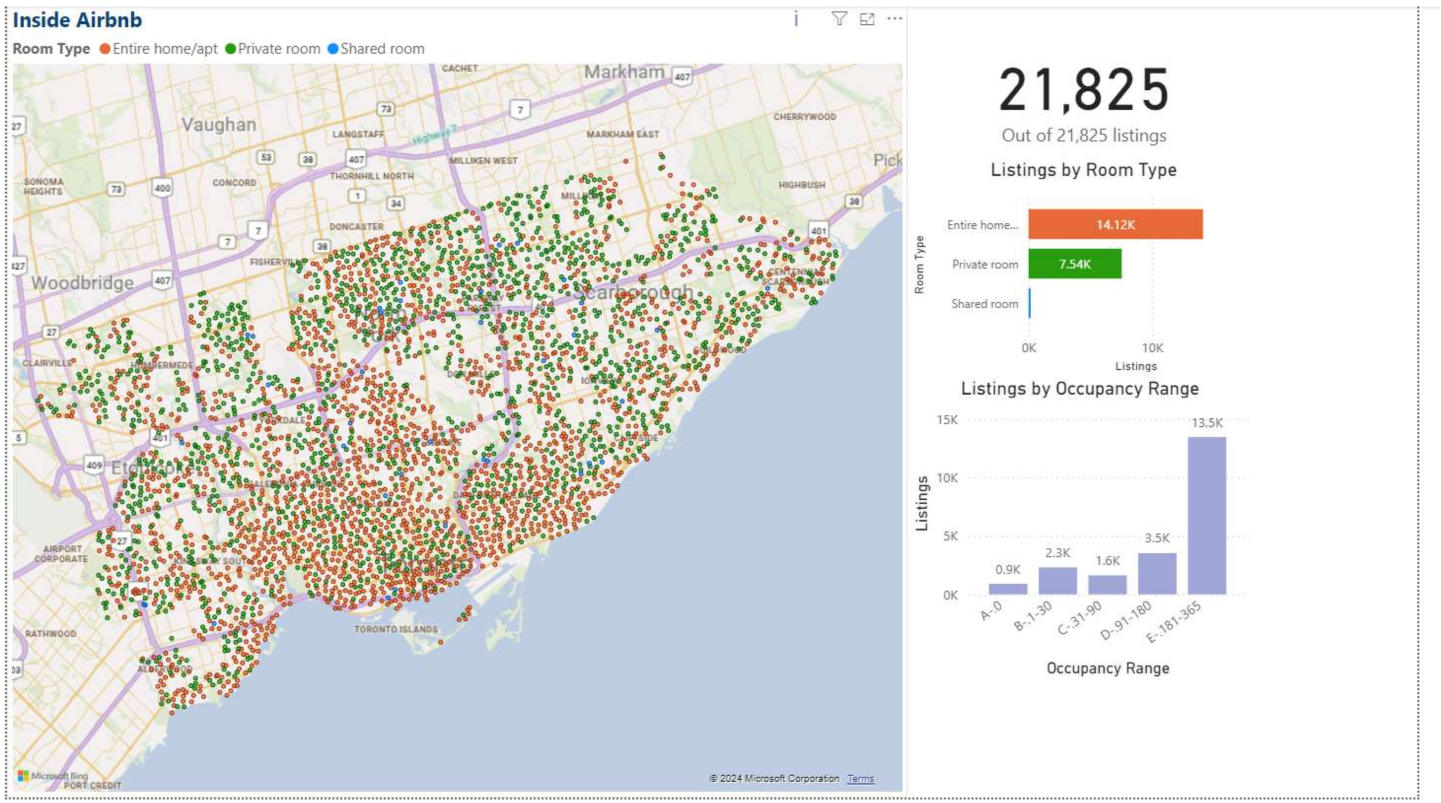
Next, the required tables are selected, and the data is prepared for the creation of visualizations. This process includes cleaning, transforming, and structuring the information to ensure its proper analysis use.



Subsequently, the previously identified relationships between the selected tables are modelled. This step is crucial to ensure data integrity and enable efficient analysis in the visualizations.

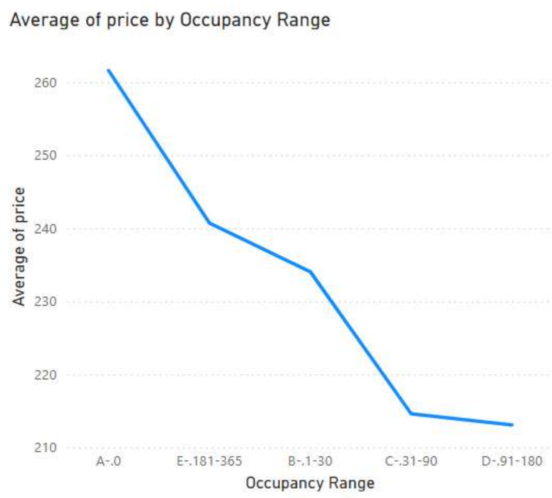


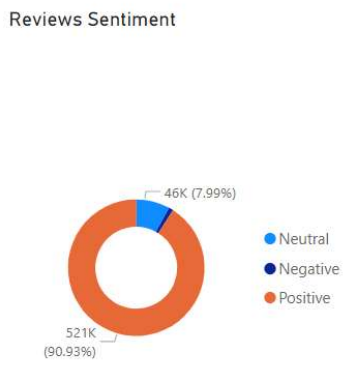
Finally, several visualizations are created. In some cases, to achieve this effectively, new calculated columns are necessary to extract additional insights and enrich the analysis. These columns are generated using specific formulas or transformations tailored to the business needs.



Top 20 Hosts

host_name	Entire home/apt	Private room	Shared room	Total
David	61	15		76
Sky View	59			59
Michael	42	9		51
James	28	22		50
Julie	43	5		48
Andrew	30	13	1	44
John	34	8		42
Sarah	26	14		40
Sam	31	8		39
Vladimir	20	16		36
Daniel	22	13		35
Matthew	25	9		34
Peter	25	9		34
Paul	30	2		32
Alex	20	9	1	30
Anna	26	4		30
Maria	23	7		30
Jonathan	17	12		29
Mark	24	5		29
Sophie	20	9		29
Total	606	189	2	797





Data Analysis and Exploration

Business Questions Addressed

The main business questions addressed through this analysis include:

1. **What factors influence Airbnb pricing in Toronto?**
2. **How does sentiment affect customer ratings for listings?**
3. **What are the trends in listing availability across different seasons?**
4. **How do hosts' response rates correlate with customer satisfaction?**

Key Insights from Data

- **Sentiment Analysis**: Listings with more positive reviews tend to have higher occupancy rates. Additionally, hosts with higher response rates often receive more positive sentiment in reviews.
- **Pricing Trends**: Price patterns were observed across different neighborhoods and room types. Listings with more amenities and better locations were generally more expensive.
- **Availability Trends**: There was a notable difference in listing availability based on the season, with more listings being available in the summer months.

Data Visualization with Power BI

To communicate these insights, we created several visualizations using **Power BI**:

- **Price Distribution**: A histogram showing the distribution of Airbnb listing prices across various neighborhoods.
- **Sentiment Distribution**: A pie chart showing the proportion of positive, negative, and neutral sentiments across all reviews.
- **Seasonal Availability Trends**: A line chart displaying the number of days listings were available across different months in 2023.

These visualizations helped to illustrate key trends in a clear and interactive format, supporting informed decision-making.

Final Deliverables

The final deliverables for this project include:

1. **ETL Pipeline**: A fully functional ETL pipeline built using **SSIS**, which extracts, transforms, and loads data into SQL Server.
2. **SQL Server Database**: A structured database containing clean and transformed data.
3. **Data Visualizations**: Insights and trends presented through interactive **Power BI** dashboards.
4. **SSMS**
5. **Reports**: A detailed report summarizing the project's methodology, insights, and visualizations.
6. **Video Demonstration**: A video recording demonstrating the execution and functionality of the pipeline and the visualizations.

CONCLUSION

This project successfully demonstrated the ability to process and analyze large datasets using **big data tools** and **ETL pipelines**. By leveraging MongoDB, SQL Server, Power BI, and SSIS, we created a pipeline that transformed raw data into valuable insights for Airbnb business decisions. The visualizations provide actionable insights into pricing trends, customer sentiment, and listing availability, which can inform strategic decisions for Airbnb hosts and stakeholders.