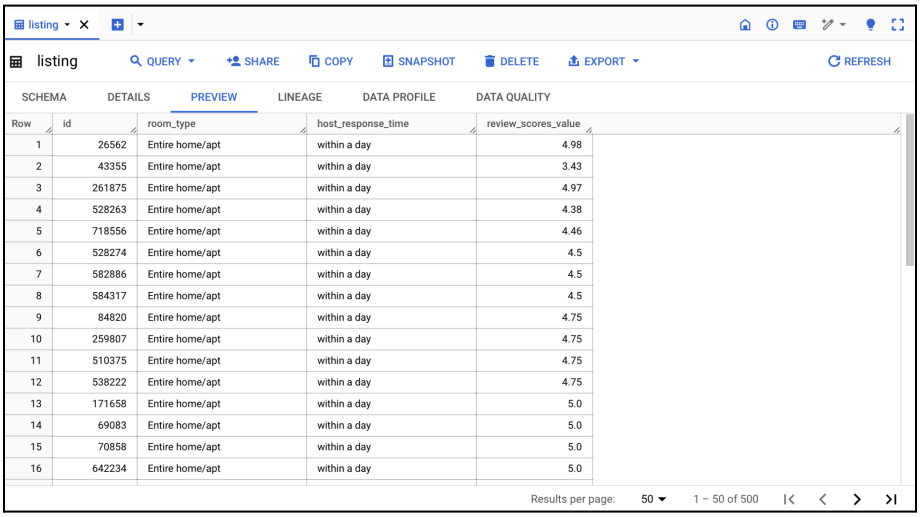
Project AirBnb

We have 2 data sets at our disposal:

- Listing
- Calendar

In the “**listing**” table, we have one line per rental:

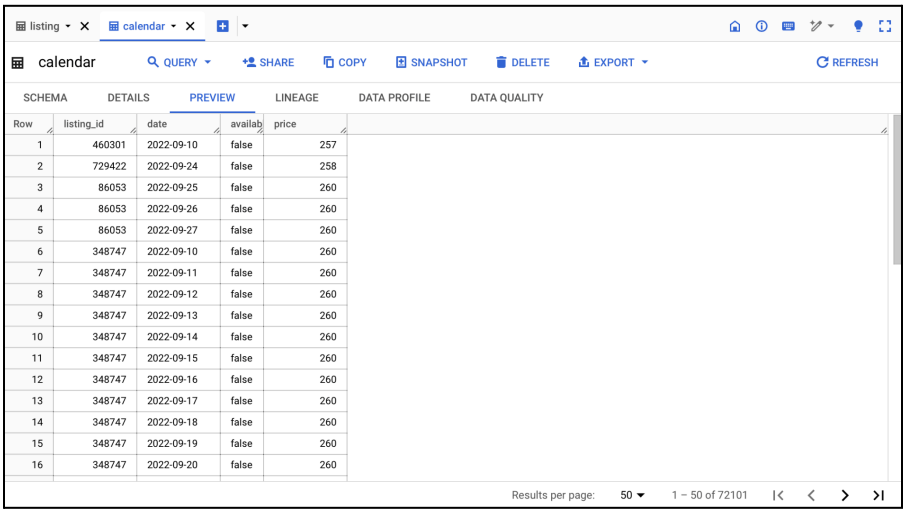
- id - the ID of the rental
- room_type - the type of rental (e.g., entire home, private room)
- host_response_time - the average response time displayed in Airbnb to customers
- review_scores_value - the average review score for the rental In the calendar table, we have one entry per rental and per day



Row	id	room_type	host_response_time	review_scores_value
1	26562	Entire home/apt	within a day	4.98
2	43355	Entire home/apt	within a day	3.43
3	261875	Entire home/apt	within a day	4.97
4	528263	Entire home/apt	within a day	4.38
5	718556	Entire home/apt	within a day	4.46
6	528274	Entire home/apt	within a day	4.5
7	582886	Entire home/apt	within a day	4.5
8	584317	Entire home/apt	within a day	4.5
9	84820	Entire home/apt	within a day	4.75
10	259807	Entire home/apt	within a day	4.75
11	510375	Entire home/apt	within a day	4.75
12	538222	Entire home/apt	within a day	4.75
13	171658	Entire home/apt	within a day	5.0
14	69083	Entire home/apt	within a day	5.0
15	70858	Entire home/apt	within a day	5.0
16	642234	Entire home/apt	within a day	5.0

In the “**calendar**” table, we have one line per rental:

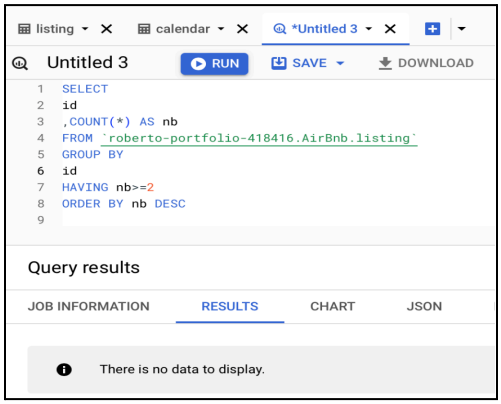
- listing_id - the ID of the rental
- date - the date of the rental or non-rental
- available - a boolean indicating the availability of the rental
- price - the price for the specific date



Row	listing_id	date	availab	price
1	460301	2022-09-10	false	257
2	729422	2022-09-24	false	258
3	86053	2022-09-25	false	260
4	86053	2022-09-26	false	260
5	86053	2022-09-27	false	260
6	348747	2022-09-10	false	260
7	348747	2022-09-11	false	260
8	348747	2022-09-12	false	260
9	348747	2022-09-13	false	260
10	348747	2022-09-14	false	260
11	348747	2022-09-15	false	260
12	348747	2022-09-16	false	260
13	348747	2022-09-17	false	260
14	348747	2022-09-18	false	260
15	348747	2022-09-19	false	260
16	348747	2022-09-20	false	260

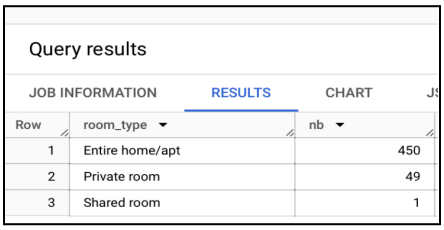
1. Exploring in SQL the ‘listing’ table on Big Query

Primary Key for ‘listing’ table is ‘id’




```
SELECT
id
,COUNT(*) AS nb
FROM `roberto-portfolio-418416.AirBnb.listing`
GROUP BY
id
HAVING nb>=2
ORDER BY nb DESC
```

1.1. Room type analysis

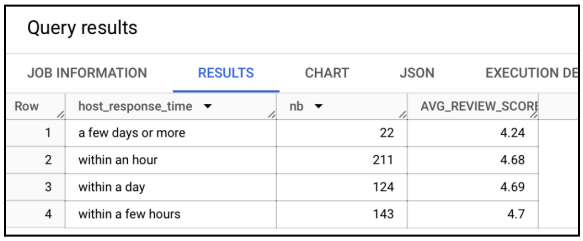


Row	room_type	nb
1	Entire home/apt	450
2	Private room	49
3	Shared room	1

```
SELECT
room_type
,COUNT(*) AS nb
FROM `roberto-portfolio-418416.AirBnb.listing`
GROUP BY room_type
ORDER BY nb DESC
```

So there are **500** rental properties in our dataset.

There are more rental announcements with “**entire home/apt**” : **90%** (450/500)



Row	host_response_time	nb	AVG_REVIEW_SCORE
1	a few days or more	22	4.24
2	within an hour	211	4.68
3	within a day	124	4.69
4	within a few hours	143	4.7

```
SELECT
host_response_time
,COUNT(*) AS nb
,ROUND(AVG(review_scores_value),2)
AVG_REVIEW_SCORE
FROM `roberto-portfolio-418416.AirBnb.listing`
GROUP BY host_response_time
ORDER BY AVG_REVIEW_SCORE
```

96% of rentals respond in less than a day

- Is there a correlation between host response time and review score ?
- We could create an 'NPS' with the review score to categorize respondents into detractors, neutrals and promoters.

2. Exploring in SQL the ‘calendar’ table

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EX

Row	listing_id	nb
1	434658	145
2	729482	145
3	39666	145
4	270067	145
5	406852	145
6	517897	145
7	469712	145
8	591873	145

```
SELECT
listing_id
,COUNT(*) AS nb
FROM `roberto-portfolio-418416.AirBnb.calendar`
GROUP BY
listing_id
HAVING nb>=2
ORDER BY nb DESC
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON

Row	date	nb
1	2022-09-09	101
2	2022-09-10	500
3	2022-09-11	500
4	2022-09-12	500
5	2022-09-13	500
6	2022-09-14	500
7	2022-09-15	500

```
SELECT
date
,COUNT(*) AS nb
FROM `roberto-portfolio-418416.AirBnb.calendar`
GROUP BY
date
ORDER BY date
```

We note that there are **145 days** in the dataset, from **2022-09-09** to **2023-01-31**.

Query results

JOB INFORMATION | RESULTS | CHART | JSON

Row	available	nb	AVG_PRICE
1	true	17536	190.05
2	false	54565	147.73

```
SELECT
available
,COUNT(*) AS nb
FROM `roberto-portfolio-418416.AirBnb.calendar`
GROUP BY
available
ORDER BY available DESC
```

There are **17536 rentals available**, then **24%** (17536/72101), while **54565 rentals are not available** then **76%** of the time the rentals are occupied.

We also note that prices are higher when homes are available. In particular, there is a difference of almost 45€. The average price is **158€.**

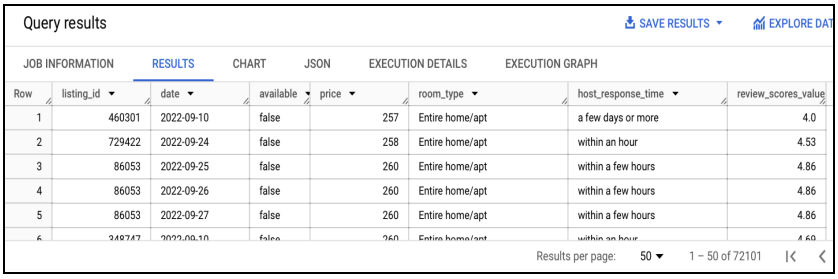
Query results

JOB INFORMATION | RESULTS | CHART | JSON

Row	AVG_PRICE
1	158.02

```
SELECT
ROUND(AVG(price),2) AVG_PRICE
FROM `roberto-portfolio-418416.AirBnb.calendar`
```

3. Joining the 'Listing' and 'Calendar' tables



Query results

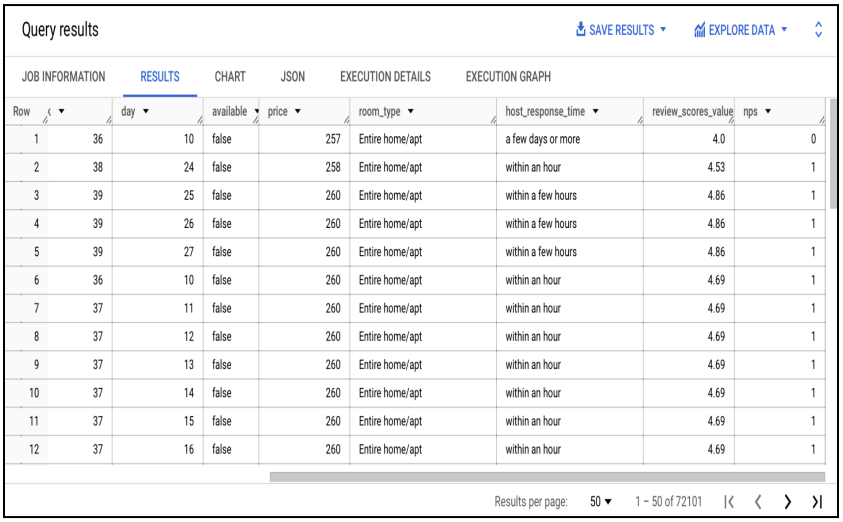
SAVE RESULTS | EXPLORE DAT

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	listing_id	date	available	price	room_type	host_response_time	review_scores_value
1	460301	2022-09-10	false	257	Entire home/apt	a few days or more	4.0
2	729422	2022-09-24	false	258	Entire home/apt	within an hour	4.53
3	86053	2022-09-25	false	260	Entire home/apt	within a few hours	4.86
4	86053	2022-09-26	false	260	Entire home/apt	within a few hours	4.86
5	86053	2022-09-27	false	260	Entire home/apt	within a few hours	4.86
6	349747	2022-09-10	false	260	Entire home/apt	within an hour	4.69

Results per page: 50 1 – 50 of 72101

```
SELECT
listing_id,
date,
available,
price,
room_type,
host_response_time,
review_scores_value
FROM `roberto-portfolio-418416.AirBnb.calendar`
as A
JOIN `roberto-portfolio-418416.AirBnb.listing` AS B
ON A.listing_id = B.id
```



Query results

SAVE RESULTS | EXPLORE DATA

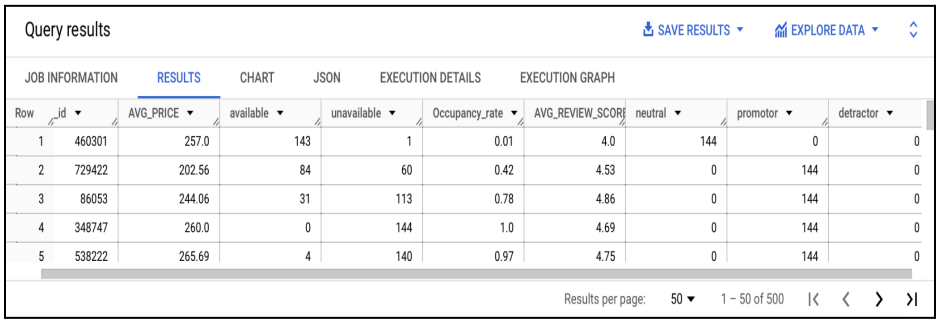
JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row		day	available	price	room_type	host_response_time	review_scores_value	nps
1	36	10	false	257	Entire home/apt	a few days or more	4.0	0
2	38	24	false	258	Entire home/apt	within an hour	4.53	1
3	39	25	false	260	Entire home/apt	within a few hours	4.86	1
4	39	26	false	260	Entire home/apt	within a few hours	4.86	1
5	39	27	false	260	Entire home/apt	within a few hours	4.86	1
6	36	10	false	260	Entire home/apt	within an hour	4.69	1
7	37	11	false	260	Entire home/apt	within an hour	4.69	1
8	37	12	false	260	Entire home/apt	within an hour	4.69	1
9	37	13	false	260	Entire home/apt	within an hour	4.69	1
10	37	14	false	260	Entire home/apt	within an hour	4.69	1
11	37	15	false	260	Entire home/apt	within an hour	4.69	1
12	37	16	false	260	Entire home/apt	within an hour	4.69	1

Results per page: 50 1 – 50 of 72101

```
SELECT
listing_id,
date,
EXTRACT(YEAR FROM date) AS year,
EXTRACT(MONTH FROM date) AS month,
EXTRACT(WEEK FROM date) AS week,
EXTRACT(DAY FROM date) AS day,
available,
price,
room_type,
host_response_time,
review_scores_value,
CASE
WHEN review_scores_value*2 >= 9 THEN 1
WHEN review_scores_value*2 <= 6 THEN -1
ELSE 0
END AS nps
FROM `roberto-portfolio-418416.AirBnb.calendar`
as A
JOIN `roberto-portfolio-418416.AirBnb.listing` AS B
ON A.listing_id = B.id
```

```
SELECT
listing_id,
ROUND(AVG(price),2) AVG_PRICE,
COUNTIF(available=true) AS available,
COUNTIF(available=false) AS unavailable,
ROUND(SAFE_DIVIDE(COUNTIF(available=false),(COUNTIF(available=true)+COUNTIF(available=false))),2) AS Occupancy_rate,
ROUND(AVG(review_scores_value),2) AVG_REVIEW_SCORE,
COUNTIF(nps=0) AS neutral,
COUNTIF(nps=1) AS promotor,
COUNTIF(nps=-1) AS detractor,
FROM `roberto-portfolio-418416.AirBnb.table_join`
GROUP BY listing_id
```



Query results

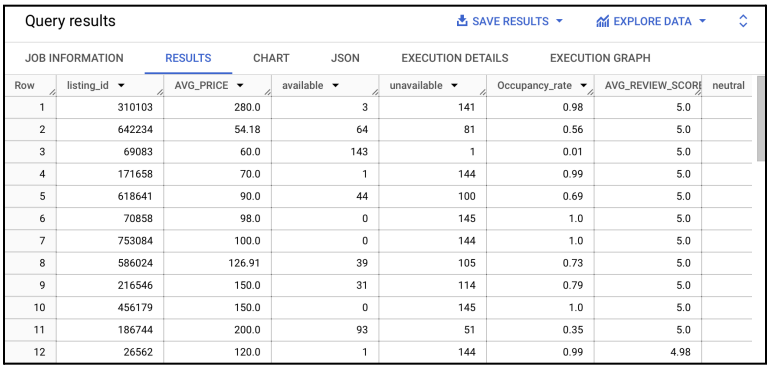
SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	_id	AVG_PRICE	available	unavailable	Occupancy_rate	AVG_REVIEW_SCORE	neutral	promotor	detractor
1	460301	257.0	143	1	0.01	4.0	144	0	0
2	729422	202.56	84	60	0.42	4.53	0	144	0
3	86053	244.06	31	113	0.78	4.86	0	144	0
4	348747	260.0	0	144	1.0	4.69	0	144	0
5	538222	265.69	4	140	0.97	4.75	0	144	0

Results per page: 50 | 1 – 50 of 500

3.1. Top 11 rentals by avg review score



Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	listing_id	AVG_PRICE	available	unavailable	Occupancy_rate	AVG_REVIEW_SCORE	neutral
1	310103	280.0	3	141	0.98	5.0	
2	642234	54.18	64	81	0.56	5.0	
3	69083	60.0	143	1	0.01	5.0	
4	171658	70.0	1	144	0.99	5.0	
5	618641	90.0	44	100	0.69	5.0	
6	70858	98.0	0	145	1.0	5.0	
7	753084	100.0	0	144	1.0	5.0	
8	586024	126.91	39	105	0.73	5.0	
9	216546	150.0	31	114	0.79	5.0	
10	456179	150.0	0	145	1.0	5.0	
11	186744	200.0	93	51	0.35	5.0	
12	26562	120.0	1	144	0.99	4.98	

```
SELECT
listing_id,
ROUND(AVG(price),2) AVG_PRICE,
COUNTIF(available=true) AS available,
COUNTIF(available=false) AS unavailable,
ROUND(SAFE_DIVIDE(COUNTIF(available=false),(C
OUNTIF(available=true)+COUNTIF(available=false))),
2) AS Occupancy_rate,
ROUND(AVG(review_scores_value),2)
AVG_REVIEW_SCORE,
COUNTIF(nps=0) AS neutral,
COUNTIF(nps=1) AS promotor,
COUNTIF(nps=-1) AS detractor,
FROM `roberto-portfolio-418416.AirBnb.table_join`
GROUP BY listing_id
order by AVG_REVIEW_SCORE DESC
```

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	date	AVG_PRICE	available	unavailable	Occupancy_rate	AVG_REVIEW_SCORE	neutral
1	2022-09-09	153.45	3	98	0.97	4.72	10
2	2022-09-10	157.98	16	484	0.97	4.67	67
3	2022-09-11	155.68	26	474	0.95	4.67	67
4	2022-09-12	155.84	30	470	0.94	4.67	67
5	2022-09-13	155.8	26	474	0.95	4.67	67
6	2022-09-14	156.08	29	471	0.94	4.67	67
7	2022-09-15	158.16	30	470	0.94	4.67	67
8	2022-09-16	160.39	33	467	0.93	4.67	67
9	2022-09-17	160.57	36	464	0.93	4.67	67
10	2022-09-18	158.27	40	460	0.92	4.67	67
11	2022-09-19	158.11	39	461	0.92	4.67	67
12	2022-09-20	158.18	40	460	0.92	4.67	67

Results per page: 50 | 1 – 50 of 145

```
SELECT
date,
ROUND(AVG(price),2) AVG_PRICE,
COUNTIF(available=true) AS available,
COUNTIF(available=false) AS unavailable,
ROUND(SAFE_DIVIDE(COUNTIF(available=false),(COUNTIF(available=true)+COUNTIF(available=false))),2) AS Occupancy_rate,
ROUND(AVG(review_scores_value),2) AVG_REVIEW_SCORE,
COUNTIF(nps=0) AS neutral,
COUNTIF(nps=1) AS promotor,
COUNTIF(nps=-1) AS detractor,
FROM
`roberto-portfolio-418416.AirBnb.table_join`
GROUP BY date
```

4. Python analysis on Google Colab

Analysis AirBnb.ipynb

In this Analysis we will answer the following problems:

- Is the average price difference between "entire home" and "private room" properties significant?
- Does a host's response time affect its final review score?
- Price trend analysis: are hosts taking advantage of the calendar to set their prices?

The objective will be to load the data, clean and format it, before exploring

and analyzing it.

We will trying to answer the following problems:

- Q1 : Is the average price difference between "entire home" and "private room" properties significant?
- Q2: Does a host's response time affect its final review score?
- Q3 :Price trend analysis: are hosts taking advantage of the calendar to set their prices?

Data Description

Data collected on the Airbnb website on September 9, 2022.

Detailed description:

- **calendar**: Table gathering reservation calendar information. Contains prices and availability for coming year.
  - *listing_id*: id of the accommodation
  - *date*: day of possible reservation
  - *available*: availability/unavailability of the accommodation
  - *price*: price according to the day
- **listing**: Table with an overview of the accommodations.
  - *id* : Airbnb's unique identifier for the listing (accomodation)
  - *host_response_time* : The average response time of the host for this accommodation
  - *room_type* : Entire home/apt | Private room | Shared room
  - *review_scores_value* : The average reviews score that the listing has

4.1. Importing datasets

```
[1] import pandas as pd
```

```
[2] df_calendar = pd.read_csv("calendar.csv")
```

```
[3] df_calendar.head(10)
```

	listing_id	date	available	price
0	130420	2022-09-10	f	$220.00
1	130420	2022-09-11	f	$210.00
2	130420	2022-09-12	t	$210.00
3	130420	2022-09-13	t	$210.00
4	130420	2022-09-14	t	$210.00
5	130420	2022-09-15	f	$210.00
6	130420	2022-09-16	f	$220.00
7	130420	2022-09-17	f	$220.00
8	130420	2022-09-18	f	$210.00
9	130420	2022-09-19	f	$210.00

Next steps: View recommended plots

```
[4] df_listing = pd.read_csv("listing.csv")
```

```
[5] df_listing.head(10)
```

	id	room_type	host_response_time	review_scores_value
0	130420	Entire home/apt	within a few hours	4.43
1	23441	Entire home/apt	within a day	4.64
2	5396	Entire home/apt	within an hour	4.54
3	7397	Entire home/apt	within an hour	4.72
4	24260	Entire home/apt	within a few hours	4.64
5	137112	Entire home/apt	within an hour	4.61
6	26562	Entire home/apt	within a day	4.98
7	137857	Entire home/apt	within an hour	4.55
8	138548	Entire home/apt	within a few hours	4.79
9	139074	Entire home/apt	within a day	4.85

Next steps: View recommended plots

4.2. Data cleaning and formatting

Date

```
[6] # The "date" column is formatted as a "string" then we use the to_datetime function, to transform the "date" column to datetime format

    df_calendar['date'] = pd.to_datetime(df_calendar['date'])
```

```
[7] # We have also added a month column

    df_calendar['month']=df_calendar['date'].dt.month
```

```
[8] df_calendar.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 72101 entries, 0 to 72100
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   listing_id  72101 non-null  int64
 1   date        72101 non-null  datetime64[ns]
 2   available   72101 non-null  object
 3   price       72101 non-null  object
 4   month       72101 non-null  int64
dtypes: datetime64[ns](1), int64(2), object(2)
memory usage: 2.8+ MB
```

```
[9] df_calendar.head()
```

	listing_id	date	available	price	month
0	130420	2022-09-10	f	$220.00	9
1	130420	2022-09-11	f	$210.00	9
2	130420	2022-09-12	t	$210.00	9
3	130420	2022-09-13	t	$210.00	9
4	130420	2022-09-14	t	$210.00	9

Next steps: View recommended plots

Price

```
[10] # The "price" column is in a particular format
     # We will create a function clean_price, which takes as input a variable "price" in the format seen above, and returns the price in float format.

     def clean_price(price):
       price_clean = price.replace('$', '').replace(",",'')
       price_clean = float(price_clean)
       return price_clean
```

```
df_calendar["price"] = df_calendar["price"].apply(clean_price)
df_calendar["price"]
```

```
0        220.0
1        210.0
2        210.0
3        210.0
4        210.0
         ...
72096    100.0
72097    100.0
72098    100.0
72099    100.0
72100    100.0
Name: price, Length: 72101, dtype: float64
```

4.2. Data exploration

The objective of this part is to discover the data and answer the questions.

```
[12] import plotly.express as px
```

```
[17] # Oldest and most recent reservation date.

     oldest = df_calendar['date'].min()
     most_recent = df_calendar['date'].max()
     f'The oldest reservation date is {oldest} and the most recent is {most_recent} '
```

```
'The oldest reservation date is 2022-09-09 00:00:00 and the most recent is 2023-01-31 00:00:00 '
```

```
[14] # Number of unique "listing_id"

     len(pd.unique(df_calendar['listing_id']))
```

```
500
```

```
[15] # Descriptive statistics of price column

     df_calendar['price'].describe()
```

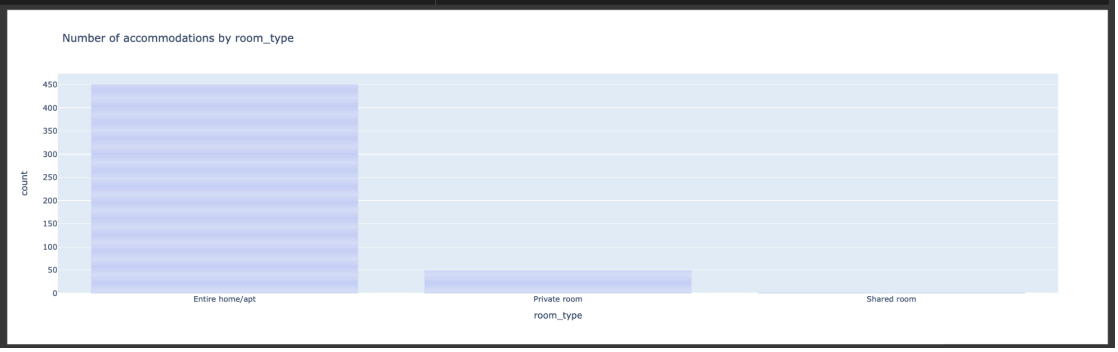
```
count    72101.000000
mean       158.020041
std        142.359070
min         30.000000
25%         85.000000
50%        120.000000
75%        175.000000
max       4000.000000
Name: price, dtype: float64
```

```
[ ] # We note that the average price is 158 $ for one night, and that this price varies from 30 $ to 4 000$.
```

```
# Percentage of days an accommodation is available on average.

percent_available = (df_calendar[df_calendar['available']=='t'].count()/(df_calendar[df_calendar['available']=='f'].count()+df_calendar[df_calendar['available']=='t'].count())).listing_id

f'The average percent of days an accomodation is available is {round(percent_available*100)}% '
```

```
'The average percent of days an accomodation is available is 24% '
```

```
[25] fig = px.bar(df_listing, x='room_type', title="Number of accommodations by room_type")
     fig.show()
```



Q1 : Is the average review score different between "entire home" and "private room" properties? Is it significant?

To answer this question, we propose to perform a z-test.
We use a z-test because we want to compare two averages and to determine whether they are related. We do not use a t-test because our sample n>30.

```
[ ] from statsmodels.stats.weightstats import ztest
```

With the table listing, we create two Dataframe:

- home : listing with room_type "Entire home/apt"
- private_room : listing with room_type "Private room"

```
private_room = df_listing[df_listing['room_type']=="Private room"]
private_room.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 49 entries, 13 to 498
Data columns (total 4 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   id                   49 non-null     int64
 1   room_type            49 non-null     object
 2   host_response_time   49 non-null     object
 3   review_scores_value  49 non-null     float64
dtypes: float64(1), int64(1), object(2)
memory usage: 1.9+ KB
```

```
[ ] home = df_listing[df_listing['room_type']=="Entire home/apt"]
    home.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 450 entries, 0 to 499
Data columns (total 4 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   id                   450 non-null    int64
 1   room_type            450 non-null    object
 2   host_response_time   450 non-null    object
 3   review_scores_value  450 non-null    float64
dtypes: float64(1), int64(1), object(2)
memory usage: 17.6+ KB
```

```
[ ] z_score, p_value = ztest(home['review_scores_value'],private_room['review_scores_value'])
    print(p_value)
```

```
0.047917748274801955
```

The z-test is significant because the p-value is less than 5%. We can therefore validate the hypothesis that there is a significant difference in the average review score between 'Entire home' and 'Private room'.

Q2: Does a host's response time affect its final review score?

```
[34] # The proportion of accommodations that have a host with the longest response time "a few days or more "

     longest_response_time = len(df_listing[df_listing['host_response_time']=="a few days or more"])
     longest_response_time
```

```
22
```

```
[35] all_response = len(df_listing)
     all_response
```

```
500
```

```
[37] proportion_longest_response_time = longest_response_time /all_response
     f'The proportion of accommodations that have a host with the longest response time is {round(proportion_longest_response_time*100)}% '
```

```
'The proportion of accommodations that have a host with the longest response time is 4% '
```

The **proportion of accommodations** that have a host with the longest response time is **4%**

Calculation of the correlation between host_response_time and review_scores_value, to see if the two variables are correlated.

First, we create a new column "host_response_time_num", which contains the value of host_response_time in numerical values with the following rule:

- 1 : within an hour
- 2 : within a few hours
- 3 : within a day
- 4 : a few days or more

```
[38] dict_response_time= {
        'within an hour': 1,
        'within a few hours': 2,
        'within a day': 3,
        'a few days or more': 4
    }
```

```
[40] df_listing['host_response_time_num'] = df_listing['host_response_time'].replace(dict_response_time)
    print(df_listing['host_response_time_num'].unique())
    print(df_listing['host_response_time'].unique())
```

```
[2 3 1 4]
['within a few hours' 'within a day' 'within an hour' 'a few days or more']
```

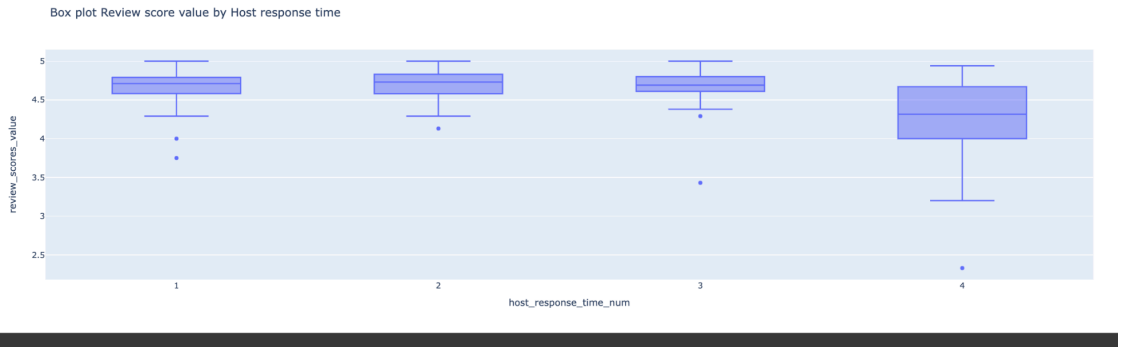
```
[41] df_listing['host_response_time_num'].corr(df_listing['review_scores_value'])
```

```
-0.18039981452312104
```

There is a negative correlation between the variables "host_response_time_num" and "review_scores_value".

This means that when 'host_response_time' increases, the 'review score value' decreases. Then, we can conclude that the host response time influences the review score value from the customers.

```
[44] fig = px.box(df_listing, x='host_response_time_num', y='review_scores_value', title = 'Box plot Review score value by Host response time')
    fig.show()
```



In this box plot, the review score's median is lower when hosts take longer than one day to respond.

Q3: Pricing trend analysis: are hosting companies taking advantage of the calendar to set their prices?

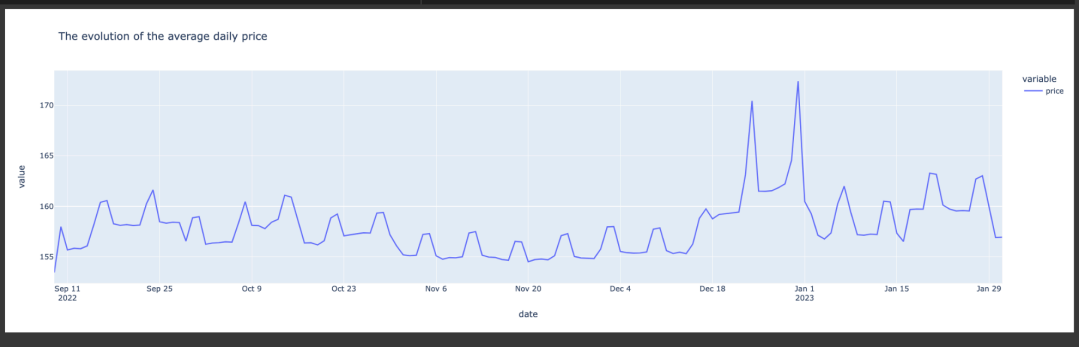
We wonder if hosts take advantage of the calendar to set their prices. For example higher rates during national holidays.
We assume that Airbnb hosts set their own prices.

```
[45] # We put the "date" column in index

     df_calendar.set_index('date',inplace=True)
```

```
[46] # We stored the average price per day in a mean_price variable

     mean_price = df_calendar['price'].resample('D').mean()
     mean_price
```

```
date
2022-09-09    153.445545
2022-09-10    157.982000
2022-09-11    155.676000
2022-09-12    155.840000
2022-09-13    155.802000
                 ...
2023-01-27    162.688000
2023-01-28    163.012000
2023-01-29    159.994000
2023-01-30    156.910000
2023-01-31    156.942000
Freq: D, Name: price, Length: 145, dtype: float64
```

```
[47] fig = px.line(mean_price, title='The evolution of the average daily price')
     fig.show()
```



We note that the average daily price during the Christmas vacations is higher than during the rest of the year, with a jump on 24/12 and another on 31/12.

```
[48] # Percentage of properties that never change the price of their home

     std = df_calendar.groupby('listing_id')['price'].std()
     std
```

```
listing_id
5396      14.869320
7397       4.420936
9952       0.000000
10586      0.000000
10588      0.000000
             ...
784863     0.000000
786347    16.272137
786714    13.446490
787167     3.464326
788259     0.000000
Name: price, Length: 500, dtype: float64
```

```
[51] never_change_price = std[std == 0]
     never_change_price
```

```
listing_id
9952      0.0
10586     0.0
10588     0.0
11213     0.0
11487     0.0
          ...
753084    0.0
783167    0.0
784610    0.0
784863    0.0
788259    0.0
Name: price, Length: 235, dtype: float64
```

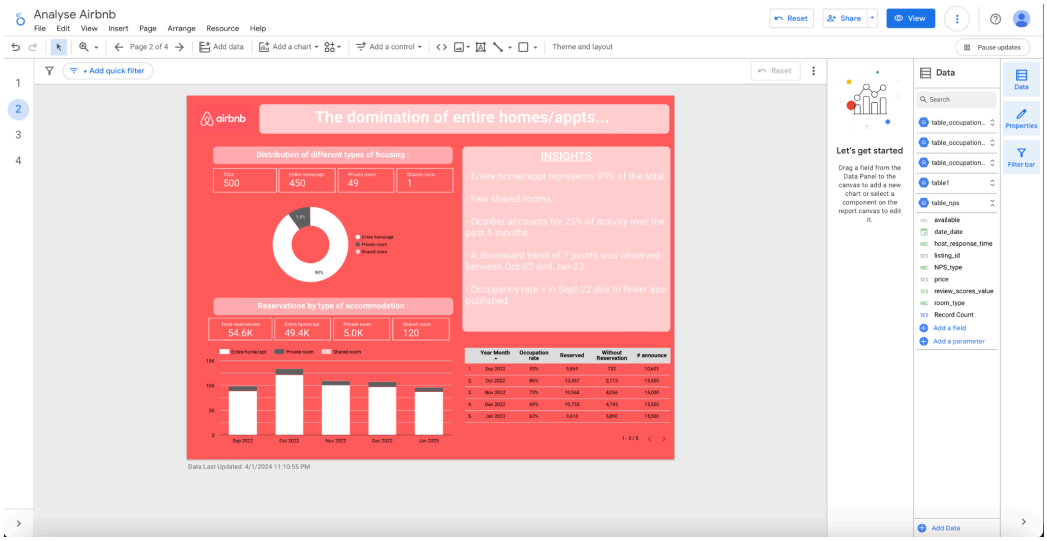
```
[56] pct_std = len(never_change_price)/len(std)
     pct_std
```

```
0.47
```

```
[57] f'The percentage of properties that never change the price of their home is {round(pct_std*100)}% '
```

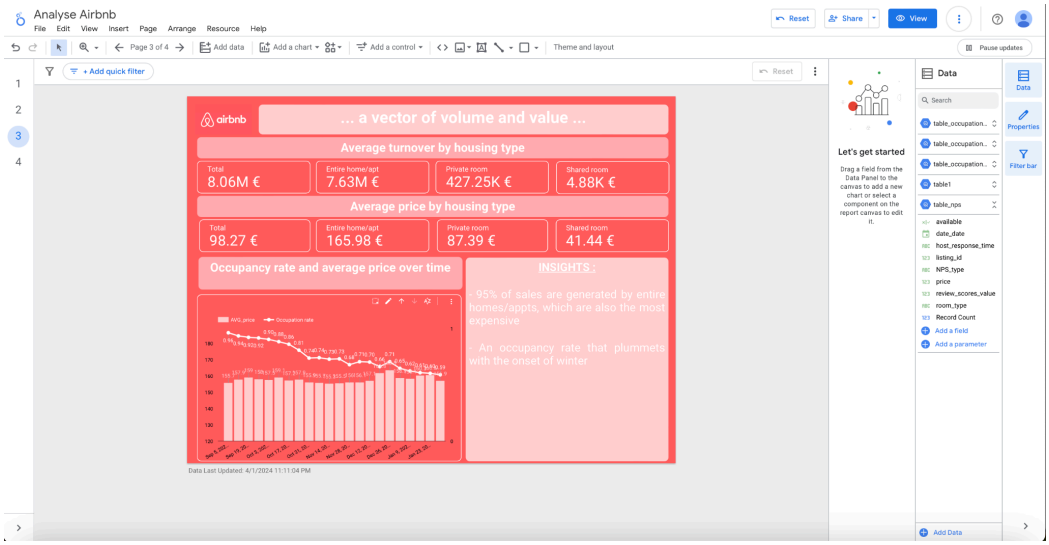
The percentage of properties that never change the price of their home is **47%**.
We can see that more than half the hosts change their price according to the calendar, then we can deduce that they take overall advantage of the calendar to set their price.

5. Dashboard on Looker Studio



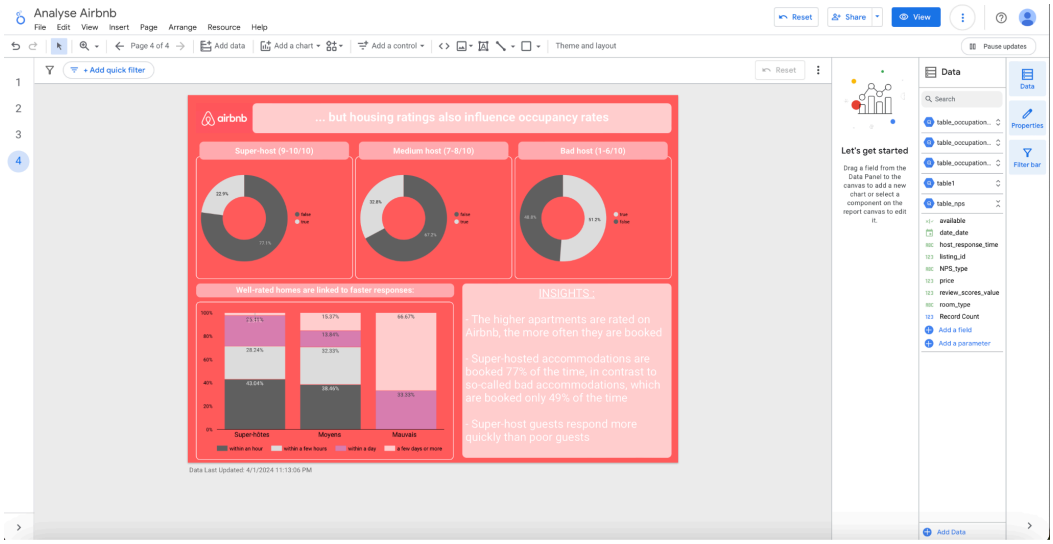
INSIGHTS :

- Entire home/appt represents 90% of the total
- Few shared rooms
- October accounts for 25% of activity over the past 5 months
- A downward trend of 7 points was observed between Oct-22 and Jan-23
- Occupancy rate > in Sept-22 due to fewer ads published



INSIGHTS :

- 95% of sales are generated by entire homes/appts, which are also the most expensive
- An occupancy rate that plummets with the onset of winter



INSIGHTS :

- The higher apartments are rated on Airbnb, the more often they are booked
- Super-hosted accommodations are booked 77% of the time, in contrast to so-called bad accommodations, which are booked only 49% of the time
- Super-host guests respond more quickly than poor guests