Data Cleaning and Join Operations

IN PYTHON

DataSet1 - Listings
DataSet2 - Reviews
DataSet3 - Calendar

```
import pandas as pd

df = pd.read_csv("C:/data science/Flats/listings.csv")
df.head()
```

	id	listing_url	scrape_id	last_scraped	name	summary	space	descri
0	241032	https://www.airbnb.com/rooms/241032	20160104002432	2016-01-04	Stylish Queen Anne Apartment	NaN	Make your self at home in this charming one-be...	Make ho cha on
1	953595	https://www.airbnb.com/rooms/953595	20160104002432	2016-01-04	Bright & Airy Queen Anne Apartment	Chemically sensitive? We've removed the irrita...	Beautiful, hypoallergenic apartment in an extr...	Chem sens rem the i
2	3308979	https://www.airbnb.com/rooms/3308979	20160104002432	2016-01-04	New Modern House-Amazing water view	New modern house built in 2013. Spectacular s...	Our house is modern, light and fresh with a wa...	m house in Specta
3	7421966	https://www.airbnb.com/rooms/7421966	20160104002432	2016-01-04	Queen Anne Chateau	A charming apartment that sits atop Queen Anne...	NaN	A cha apar th atop Q A
4	278830	https://www.airbnb.com/rooms/278830	20160104002432	2016-01-04	Charming craftsman 3 bdm house	Cozy family craftman house in beautiful neighb...	Cozy family craftman house in beautiful neighb...	Cozy cra ho bea nei

5 rows × 92 columns

```
# number of rows and columns
df.shape
```

```
(3818, 92)
```

```
# get all columns from dataset
df.columns
```

```
Index(['id', 'listing_url', 'scrape_id', 'last_scraped', 'name', 'summary',
       'space', 'description', 'experiences_offered', 'neighborhood_overview',
       'notes', 'transit', 'thumbnail_url', 'medium_url', 'picture_url',
       'xl_picture_url', 'host_id', 'host_url', 'host_name', 'host_since',
       'host_location', 'host_about', 'host_response_time',
       'host_response_rate', 'host_acceptance_rate', 'host_is_superhost',
       'host_thumbnail_url', 'host_picture_url', 'host_neighbourhood',
       'host_listings_count', 'host_total_listings_count',
       'host_verifications', 'host_has_profile_pic', 'host_identity_verified',
       'street', 'neighbourhood', 'neighbourhood_cleansed',
       'neighbourhood_group_cleansed', 'city', 'state', 'zipcode', 'market',
       'smart_location', 'country_code', 'country', 'latitude', 'longitude',
       'is_location_exact', 'property_type', 'room_type', 'accommodates',
       'bathrooms', 'bedrooms', 'beds', 'bed_type', 'amenities', 'square_feet',
       'price', 'weekly_price', 'monthly_price', 'security_deposit',
       'cleaning_fee', 'guests_included', 'extra_people', 'minimum_nights',
       'maximum_nights', 'calendar_updated', 'has_availability',
       'availability_30', 'availability_60', 'availability_90',
       'availability_365', 'calendar_last_scraped', 'number_of_reviews',
       'first_review', 'last_review', 'review_scores_rating',
       'review_scores_accuracy', 'review_scores_cleanliness',
       'review_scores_checkin', 'review_scores_communication',
       'review_scores_location', 'review_scores_value', 'requires_license',
       'license', 'jurisdiction_names', 'instant_bookable',
       'cancellation_policy', 'require_guest_profile_picture',
       'require_guest_phone_verification', 'calculated_host_listings_count',
       'reviews_per_month'],
      dtype='object')
```

```
# drop these columns
data = df.drop(['summary', 'experiences_offered', 'space', 'description', 'neighborhood_overview',
       'notes', 'transit', 'thumbnail_url', 'medium_url', 'picture_url',
       'xl_picture_url', 'host_url', 'host_about', 'host_thumbnail_url', 'host_picture_url', 'neighbourhood_cleansed',
       'neighbourhood_group_cleansed', 'host_has_profile_pic','license'], axis=1)
```

```
data.columns
```

```
Index(['id', 'listing_url', 'scrape_id', 'last_scraped', 'name', 'host_id',
       'host_name', 'host_since', 'host_location', 'host_response_time',
       'host_response_rate', 'host_acceptance_rate', 'host_is_superhost',
       'host_neighbourhood', 'host_listings_count',
       'host_total_listings_count', 'host_verifications',
       'host_identity_verified', 'street', 'neighbourhood', 'city', 'state',
       'zipcode', 'market', 'smart_location', 'country_code', 'country',
       'latitude', 'longitude', 'is_location_exact', 'property_type',
       'room_type', 'accommodates', 'bathrooms', 'bedrooms', 'beds',
       'bed_type', 'amenities', 'square_feet', 'price', 'weekly_price',
       'monthly_price', 'security_deposit', 'cleaning_fee', 'guests_included',
       'extra_people', 'minimum_nights', 'maximum_nights', 'calendar_updated',
       'has_availability', 'availability_30', 'availability_60',
       'availability_90', 'availability_365', 'calendar_last_scraped',
       'number_of_reviews', 'first_review', 'last_review',
       'review_scores_rating', 'review_scores_accuracy',
       'review_scores_cleanliness', 'review_scores_checkin',
       'review_scores_communication', 'review_scores_location',
       'review_scores_value', 'requires_license', 'jurisdiction_names',
       'instant_bookable', 'cancellation_policy',
       'require_guest_profile_picture', 'require_guest_phone_verification',
       'calculated_host_listings_count', 'reviews_per_month'],
      dtype='object')
```

```python
# finding null values in each column
for column in data.columns:
    null_count = data[column].isnull().sum()
    print(f"{column}:  {null_count}")
```

```
id:  0
listing_url:  0
scrape_id:  0
last_scraped:  0
name:  0
host_id:  0
host_name:  2
host_since:  2
host_location:  8
host_response_time:  523
host_response_rate:  523
host_acceptance_rate:  773
host_is_superhost:  2
host_neighbourhood:  300
host_listings_count:  2
host_total_listings_count:  2
host_verifications:  2
host_identity_verified:  2
street:  0
neighbourhood:  416
city:  0
state:  0
zipcode:  7
market:  0
smart_location:  0
country_code:  0
country:  0
latitude:  0
longitude:  0
is_location_exact:  0
property_type:  1
room_type:  0
accommodates:  0
bathrooms:  16
bedrooms:  6
beds:  1
bed_type:  0
amenities:  0
square_feet:  3721
price:  0
weekly_price:  1809
monthly_price:  2301
security_deposit:  1952
cleaning_fee:  1030
guests_included:  0
extra_people:  0
minimum_nights:  0
maximum_nights:  0
calendar_updated:  0
has_availability:  0
availability_30:  0
availability_60:  0
availability_90:  0
availability_365:  0
calendar_last_scraped:  0
number_of_reviews:  0
first_review:  627
last_review:  627
review_scores_rating:  647
review_scores_accuracy:  658
review_scores_cleanliness:  653
review_scores_checkin:  658
review_scores_communication:  651
review_scores_location:  655
review_scores_value:  656
requires_license:  0
jurisdiction_names:  0
instant_bookable:  0
cancellation_policy:  0
require_guest_profile_picture:  0
require_guest_phone_verification:  0
calculated_host_listings_count:  0
reviews_per_month:  627
```

```
# data type of each column
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3818 entries, 0 to 3817
Data columns (total 73 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   id                                3818 non-null   int64  
 1   listing_url                       3818 non-null   object 
 2   scrape_id                         3818 non-null   int64  
 3   last_scraped                      3818 non-null   object 
 4   name                              3818 non-null   object 
 5   host_id                           3818 non-null   int64  
 6   host_name                         3816 non-null   object 
 7   host_since                        3816 non-null   object 
 8   host_location                     3810 non-null   object 
 9   host_response_time                3295 non-null   object 
 10  host_response_rate                3295 non-null   object 
 11  host_acceptance_rate              3045 non-null   object 
 12  host_is_superhost                 3816 non-null   object 
 13  host_neighbourhood                3518 non-null   object 
 14  host_listings_count               3816 non-null   float64
 15  host_total_listings_count         3816 non-null   float64
 16  host_verifications                3816 non-null   object 
 17  host_identity_verified            3816 non-null   object 
 18  street                            3818 non-null   object 
 19  neighbourhood                     3402 non-null   object 
 20  city                              3818 non-null   object 
 21  state                             3818 non-null   object 
 22  zipcode                           3811 non-null   object 
 23  market                            3818 non-null   object 
 24  smart_location                    3818 non-null   object 
 25  country_code                      3818 non-null   object 
 26  country                           3818 non-null   object 
 27  latitude                          3818 non-null   float64
 28  longitude                         3818 non-null   float64
 29  is_location_exact                 3818 non-null   object 
 30  property_type                     3817 non-null   object 
 31  room_type                         3818 non-null   object 
 32  accommodates                      3818 non-null   int64  
 33  bathrooms                         3802 non-null   float64
 34  bedrooms                          3812 non-null   float64
 35  beds                              3817 non-null   float64
 36  bed_type                          3818 non-null   object 
 37  amenities                         3818 non-null   object 
 38  square_feet                       97 non-null     float64
 39  price                             3818 non-null   object 
 40  weekly_price                      2009 non-null   object 
 41  monthly_price                     1517 non-null   object 
 42  security_deposit                  1866 non-null   object 
 43  cleaning_fee                      2788 non-null   object 
 44  guests_included                   3818 non-null   int64  
 45  extra_people                      3818 non-null   object 
 46  minimum_nights                    3818 non-null   int64  
 47  maximum_nights                    3818 non-null   int64  
 48  calendar_updated                  3818 non-null   object 
 49  has_availability                  3818 non-null   object 
 50  availability_30                   3818 non-null   int64  
 51  availability_60                   3818 non-null   int64  
 52  availability_90                   3818 non-null   int64  
 53  availability_365                  3818 non-null   int64  
 54  calendar_last_scraped             3818 non-null   object 
 55  number_of_reviews                 3818 non-null   int64  
 56  first_review                      3191 non-null   object 
 57  last_review                       3191 non-null   object 
 58  review_scores_rating              3171 non-null   float64
 59  review_scores_accuracy            3160 non-null   float64
 60  review_scores_cleanliness         3165 non-null   float64
 61  review_scores_checkin             3160 non-null   float64
 62  review_scores_communication       3167 non-null   float64
 63  review_scores_location            3163 non-null   float64
 64  review_scores_value               3162 non-null   float64
 65  requires_license                  3818 non-null   object 
 66  jurisdiction_names                3818 non-null   object 
 67  instant_bookable                  3818 non-null   object 
 68  cancellation_policy               3818 non-null   object 
 69  require_guest_profile_picture     3818 non-null   object 
 70  require_guest_phone_verification  3818 non-null   object 
 71  calculated_host_listings_count    3818 non-null   int64  
 72  reviews_per_month                 3191 non-null   float64
dtypes: float64(16), int64(13), object(44)
memory usage: 2.1+ MB
```

```
# replace null
columns_to_replace = ['host_listings_count', 'host_total_listings_count', 'zipcode', 'bathrooms', 'bedrooms', 'beds', 'weekly_price', 'monthly_price',
                      'security_deposit', 'cleaning_fee', 'review_scores_rating', 'review_scores_accuracy',
                      'review_scores_cleanliness', 'review_scores_checkin', 'review_scores_communication', 'review_scores_location',
                      'review_scores_value']
data[columns_to_replace] = data[columns_to_replace].fillna(0)
```

```
# changing data type to boolean
columns_to_convert = ['host_is_superhost', 'host_identity_verified', 'has_availability', 'requires_license', 'instant_bookable',
                      'require_guest_profile_picture', 'is_location_exact', 'require_guest_phone_verification']
data[columns_to_convert] = data[columns_to_convert].astype(bool)
```

```
# filling null values to unknown
columns_to_fill = ['host_response_time', 'host_neighbourhood', 'neighbourhood', 'property_type', 'host_location']
data[columns_to_fill] = data[columns_to_fill].fillna('Unknown')
```

```
# filling null values with mean value
data['reviews_per_month'].fillna(data['reviews_per_month'].mean())
```

```
0       4.070000
1       1.480000
2       1.150000
3       2.078919
4       0.890000
          ...
3813    0.300000
3814    2.000000
3815    2.078919
3816    2.078919
3817    2.078919
Name: reviews_per_month, Length: 3818, dtype: float64
```

```
# removing % and changing data type to numeric(int)
data['host_response_rate'] = data['host_response_rate'].str.rstrip('%')
data['host_response_rate'] = pd.to_numeric(data['host_response_rate'])
```

```
# removing $ and changint data type to numeric(int)
data['host_acceptance_rate'] = data['host_acceptance_rate'].str.rstrip('%')
data['host_acceptance_rate'] = pd.to_numeric(data['host_acceptance_rate'])
```

```
# unique values in column
data["host_response_rate"].unique()
```

```
array([ 96.,  98.,  67.,  nan, 100.,  71.,  97.,  60.,  50.,  31.,  90.,
        70.,  88.,  80.,  63.,  33.,  99.,  75.,  83.,  94.,  58.,  43.,
        93.,  92.,  40.,  57.,  89.,  95.,  78.,  81.,  91.,  38.,  86.,
        30.,  56.,  76.,  64.,  82.,  17.,  87.,  25.,  69.,  53.,  65.,
        68.,  55.])
```

```
data["city"].unique()
```

```
array(['Seattle', 'Ballard, Seattle', 'West Seattle', 'Seattle ', '西雅图',
       'Phinney Ridge Seattle', 'seattle'], dtype=object)
```

```
# filling null values with mean value
data['host_response_rate'].fillna(data['host_response_rate'].mean())
data['host_acceptance_rate'].fillna(data['host_acceptance_rate'].mean())
```

```
0       100.000000
1       100.000000
2       100.000000
3        99.967159
4        99.967159
           ...
3813    100.000000
3814    100.000000
3815     99.967159
3816     99.967159
3817     99.967159
Name: host_acceptance_rate, Length: 3818, dtype: float64
```

```
# dropping rows where column_name is null
Data = data.dropna(subset=['host_name'])
Data.shape
```

```
(3816, 73)
```

```
Data.dropna(subset=['host_since'])
Data.shape
```

```
(3816, 73)
```

```
# duplicate values in dataset
Data.duplicated().sum()
```

```
np.int64(0)
```

```
Data = Data.sort_values(by='first_review')
Data['first_review'] = Data['first_review'].bfill()
Data = Data.reset_index(drop=True)
```

```
Data = Data.sort_values(by='last_review')
Data['last_review'] = Data['last_review'].bfill()
Data = Data.reset_index(drop=True)
```

```
Data['first_review'] = pd.to_datetime(Data['first_review'])
Data['last_review'] = pd.to_datetime(Data['last_review'])
```

```
Data = Data.sort_values(by='host_since')
Data['host_since'] = Data['host_since'].ffill()
Data = Data.reset_index(drop=True)
Data['host_since'] = pd.to_datetime(Data['host_since'])
```

```
Data['first_review'].unique()
```

```
<DatetimeArray>
['2015-08-16 00:00:00', '2015-06-08 00:00:00',                 'NaT',
 '2013-09-30 00:00:00', '2015-07-27 00:00:00', '2015-07-07 00:00:00',
 '2010-03-21 00:00:00', '2015-03-22 00:00:00', '2009-07-17 00:00:00',
 '2011-06-15 00:00:00',
 ...
 '2015-05-05 00:00:00', '2015-12-27 00:00:00', '2015-08-27 00:00:00',
 '2015-06-06 00:00:00', '2015-11-03 00:00:00', '2015-12-19 00:00:00',
 '2015-11-11 00:00:00', '2015-10-09 00:00:00', '2015-12-18 00:00:00',
 '2015-11-18 00:00:00']
Length: 985, dtype: datetime64[ns]
```

```
Data['last_review'].unique()
```

```
<DatetimeArray>
['2015-09-30 00:00:00', '2015-09-28 00:00:00',                 'NaT',
 '2015-12-02 00:00:00', '2015-12-01 00:00:00', '2015-08-11 00:00:00',
 '2016-01-03 00:00:00', '2015-12-14 00:00:00', '2015-12-09 00:00:00',
 '2015-12-28 00:00:00',
 ...
 '2015-02-11 00:00:00', '2015-06-29 00:00:00', '2015-04-23 00:00:00',
 '2015-04-29 00:00:00', '2015-05-27 00:00:00', '2015-06-25 00:00:00',
 '2015-07-09 00:00:00', '2015-09-04 00:00:00', '2015-07-11 00:00:00',
 '2015-11-26 00:00:00']
Length: 322, dtype: datetime64[ns]
```

```
Data['host_since'].unique()
```

```
<DatetimeArray>
['2008-11-10 00:00:00', '2009-01-08 00:00:00', '2009-02-16 00:00:00',
 '2009-03-03 00:00:00', '2009-03-30 00:00:00', '2009-04-26 00:00:00',
 '2009-05-30 00:00:00', '2009-06-09 00:00:00', '2009-08-09 00:00:00',
 '2009-08-10 00:00:00',
 ...
 '2015-12-20 00:00:00', '2015-12-21 00:00:00', '2015-12-22 00:00:00',
 '2015-12-27 00:00:00', '2015-12-28 00:00:00', '2015-12-29 00:00:00',
 '2015-12-30 00:00:00', '2016-01-01 00:00:00', '2016-01-02 00:00:00',
 '2016-01-03 00:00:00']
Length: 1380, dtype: datetime64[ns]
```

```
# reading dataset2
df2 = pd.read_csv("C:/data science/Flats/reviews.csv")
df2.head()
```

	listing_id	id	date	reviewer_id	reviewer_name	comments
0	7202016	38917982	2015-07-19	28943674	Bianca	Cute and cozy place. Perfect location to every...
1	7202016	39087409	2015-07-20	32440555	Frank	Kelly has a great room in a very central locat...
2	7202016	39820030	2015-07-26	37722850	Ian	Very spacious apartment, and in a great neighb...
3	7202016	40813543	2015-08-02	33671805	George	Close to Seattle Center and all it has to offe...
4	7202016	41986501	2015-08-10	34959538	Ming	Kelly was a great host and very accommodating ...

```
# number of rows and columns for dataset2
df2.shape
```

```
(84849, 6)
```

```
# droping column
data2 = df2.drop(['comments'], axis=1)
```

```
# column name of dataset2
data2.columns
```

```
Index(['listing_id', 'id', 'date', 'reviewer_id', 'reviewer_name'], dtype='object')
```

```
# changing data type to datetime
data2['date'] = pd.to_datetime(data2['date'])
```

```
# data type of dataset2
data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 84849 entries, 0 to 84848
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   listing_id     84849 non-null  int64
 1   id             84849 non-null  int64
 2   date           84849 non-null  datetime64[ns]
 3   reviewer_id    84849 non-null  int64
 4   reviewer_name  84849 non-null  object
dtypes: datetime64[ns](1), int64(3), object(1)
memory usage: 3.2+ MB
```

```
# reading dataset3
df3 = pd.read_csv("C:/data science/Flats/calendar.csv")
df3.head()
```

	listing_id	date	available	price
0	241032	2016-01-04	t	$85.00
1	241032	2016-01-05	t	$85.00
2	241032	2016-01-06	f	NaN
3	241032	2016-01-07	f	NaN
4	241032	2016-01-08	f	NaN

```
df3.shape
```

```
(1393570, 4)
```

```
# changing data type to datetime
df3['date'] = pd.to_datetime(df3['date'])
```

```
# cahnging data type to boolean
df3['available'] = df3['available'].astype(bool)
```

```
df3['price'] = df3['price'].str.lstrip('$') # Remove $
```

```
df3['price'] = df3['price'].str.replace(',', '')   # Remove commas
```

```
df3.head()
```

	listing_id	date	available	price
0	241032	2016-01-04	True	85.00
1	241032	2016-01-05	True	85.00
2	241032	2016-01-06	True	NaN
3	241032	2016-01-07	True	NaN
4	241032	2016-01-08	True	NaN

```
df3['price'] = pd.to_numeric(df3['price'])    # Convert to numeric
```

```
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1393570 entries, 0 to 1393569
Data columns (total 4 columns):
 #   Column      Non-Null Count    Dtype
---  ------      --------------    -----
 0   listing_id  1393570 non-null  int64
 1   date        1393570 non-null  datetime64[ns]
 2   available   1393570 non-null  bool
 3   price       934542 non-null   float64
dtypes: bool(1), datetime64[ns](1), float64(1), int64(1)
memory usage: 33.2 MB
```

```
df3['price'] = df3['price'].fillna(df3['price'].mean())
```

```
# data types of dataset3
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1393570 entries, 0 to 1393569
Data columns (total 4 columns):
 #   Column      Non-Null Count    Dtype
---  ------      --------------    -----
 0   listing_id  1393570 non-null  int64
 1   date        1393570 non-null  datetime64[ns]
 2   available   1393570 non-null  bool
 3   price       1393570 non-null  float64
dtypes: bool(1), datetime64[ns](1), float64(1), int64(1)
memory usage: 33.2 MB
```

```
df3.columns
```

```
Index(['listing_id', 'date', 'available', 'price'], dtype='object')
```

```
# performing outer join
outer_join = pd.merge(data2, df3, on = 'listing_id', how = 'outer')
print("Outer Join:\n", outer_join)
```

```
Outer Join:
          listing_id  id date_x  reviewer_id reviewer_name     date_y  \
0               3335 NaN    NaT          NaN           NaN 2016-01-04
1               3335 NaN    NaT          NaN           NaN 2016-01-05
2               3335 NaN    NaT          NaN           NaN 2016-01-06
3               3335 NaN    NaT          NaN           NaN 2016-01-07
4               3335 NaN    NaT          NaN           NaN 2016-01-08
...              ...  ..    ...          ...           ...        ...
31198735    10340165 NaN    NaT          NaN           NaN 2016-12-29
31198736    10340165 NaN    NaT          NaN           NaN 2016-12-30
31198737    10340165 NaN    NaT          NaN           NaN 2016-12-31
31198738    10340165 NaN    NaT          NaN           NaN 2017-01-01
31198739    10340165 NaN    NaT          NaN           NaN 2017-01-02

          available       price
0              True  137.944859
1              True  137.944859
2              True  137.944859
3              True  137.944859
4              True  137.944859
...             ...         ...
31198735       True   43.000000
31198736       True   43.000000
31198737       True   43.000000
31198738       True   43.000000
31198739       True   43.000000
```

```
# performing inner join
inner_join = pd.merge(data2, df3, on = 'listing_id', how = 'inner')
print("Inner Join:\n", inner_join)
```

```
Inner Join:
          listing_id        id     date_x  reviewer_id reviewer_name  \
0            7202016  38917982 2015-07-19     28943674        Bianca
1            7202016  38917982 2015-07-19     28943674        Bianca
2            7202016  38917982 2015-07-19     28943674        Bianca
3            7202016  38917982 2015-07-19     28943674        Bianca
4            7202016  38917982 2015-07-19     28943674        Bianca
...              ...       ...        ...          ...           ...
30969880     9727246  56429621 2015-12-13     30860360     Stephanie
30969881     9727246  56429621 2015-12-13     30860360     Stephanie
30969882     9727246  56429621 2015-12-13     30860360     Stephanie
30969883     9727246  56429621 2015-12-13     30860360     Stephanie
30969884     9727246  56429621 2015-12-13     30860360     Stephanie

             date_y  available       price
0        2016-01-04       True   65.000000
1        2016-01-05       True   65.000000
2        2016-01-06       True   65.000000
3        2016-01-07       True  137.944859
4        2016-01-08       True  137.944859
...             ...        ...         ...
30969880 2016-12-29       True   92.000000
30969881 2016-12-30       True   92.000000
30969882 2016-12-31       True   92.000000
30969883 2017-01-01       True   92.000000
30969884 2017-01-02       True  137.944859
```

```
# performin left join
left_join = pd.merge(data2, df3, on = 'listing_id', how = 'left')
print(left_join)
```

```
          listing_id        id     date_x  reviewer_id reviewer_name  \
0            7202016  38917982 2015-07-19     28943674        Bianca
1            7202016  38917982 2015-07-19     28943674        Bianca
2            7202016  38917982 2015-07-19     28943674        Bianca
3            7202016  38917982 2015-07-19     28943674        Bianca
4            7202016  38917982 2015-07-19     28943674        Bianca
...              ...       ...        ...          ...           ...
30969880     9727246  56429621 2015-12-13     30860360     Stephanie
30969881     9727246  56429621 2015-12-13     30860360     Stephanie
30969882     9727246  56429621 2015-12-13     30860360     Stephanie
30969883     9727246  56429621 2015-12-13     30860360     Stephanie
30969884     9727246  56429621 2015-12-13     30860360     Stephanie

             date_y  available       price
0        2016-01-04       True   65.000000
1        2016-01-05       True   65.000000
2        2016-01-06       True   65.000000
3        2016-01-07       True  137.944859
4        2016-01-08       True  137.944859
...             ...        ...         ...
30969880 2016-12-29       True   92.000000
30969881 2016-12-30       True   92.000000
30969882 2016-12-31       True   92.000000
30969883 2017-01-01       True   92.000000
30969884 2017-01-02       True  137.944859
```

```
# performing left join with selected columns
left_join = pd.merge(data2, df3, on = 'listing_id', how = 'left')

selected_columns = left_join[['listing_id', 'reviewer_name', 'price']]
print(selected_columns)
```

```
          listing_id reviewer_name       price
0            7202016        Bianca   65.000000
1            7202016        Bianca   65.000000
2            7202016        Bianca   65.000000
3            7202016        Bianca  137.944859
4            7202016        Bianca  137.944859
...              ...           ...         ...
30969880     9727246     Stephanie   92.000000
30969881     9727246     Stephanie   92.000000
30969882     9727246     Stephanie   92.000000
30969883     9727246     Stephanie   92.000000
30969884     9727246     Stephanie  137.944859

[30969885 rows x 3 columns]
```

```
# performing right join
right_join = pd.merge(data2, df3, on = 'listing_id', how = 'right')
print(right_join)
```

```
          listing_id        id     date_x  reviewer_id reviewer_name  \
0             241032  682061.0 2011-11-01     479824.0          Bro
1             241032  691712.0 2011-11-04     357699.0        Megan
2             241032  702999.0 2011-11-08    1285567.0       Marylee
3             241032  717262.0 2011-11-14     647857.0        Graham
4             241032  730226.0 2011-11-19    1389821.0        Franka
...              ...       ...        ...          ...           ...
31198735    10208623       NaN        NaT          NaN           NaN
31198736    10208623       NaN        NaT          NaN           NaN
31198737    10208623       NaN        NaT          NaN           NaN
31198738    10208623       NaN        NaT          NaN           NaN
31198739    10208623       NaN        NaT          NaN           NaN

             date_y  available       price
0        2016-01-04       True   85.000000
1        2016-01-04       True   85.000000
2        2016-01-04       True   85.000000
3        2016-01-04       True   85.000000
4        2016-01-04       True   85.000000
...             ...        ...         ...
31198735 2016-12-29       True  137.944859
31198736 2016-12-30       True  137.944859
31198737 2016-12-31       True  137.944859
31198738 2017-01-01       True  137.944859
31198739 2017-01-02       True  137.944859

[31198740 rows x 8 columns]
```

```
# performing right join with selected columns
right_join = pd.merge(data2, df3, on = 'listing_id', how = 'right')

selected_columns = right_join[['listing_id', 'reviewer_name', 'price']]
print(selected_columns)
```

```
          listing_id reviewer_name       price
0             241032          Bro   85.000000
1             241032        Megan   85.000000
2             241032       Marylee   85.000000
3             241032        Graham   85.000000
4             241032        Franka   85.000000
...              ...           ...         ...
31198735    10208623           NaN  137.944859
31198736    10208623           NaN  137.944859
31198737    10208623           NaN  137.944859
31198738    10208623           NaN  137.944859
31198739    10208623           NaN  137.944859

[31198740 rows x 3 columns]
```

Name and City of rooms having Review

```
J1 = pd.merge(Data, data2, on = 'id', how = 'inner')

selected_columns = J1[['city', 'name']].drop_duplicates()
distinct_name = selected_columns.groupby('city')['name'].unique().reset_index()

print(selected_columns)
```

```
      city                                name
0  Seattle                  condo Seattle, Wa.
1  Seattle                     Ballard Cottage
2  Seattle  Fantastic View! Close to Light Rail
3  Seattle    The Reading Room at The Farmhouse
4  Seattle      15min-Downtown! Bus-steps away!
5  Seattle    Room with Downtown Seattle Views!
6  Seattle   Modern Studio, Lovely Neighborhood
7  Seattle       Blue Nautical Room w/Breakfast
8  Seattle      Luxury Studio Convention Center
```

How does price vary with availability?

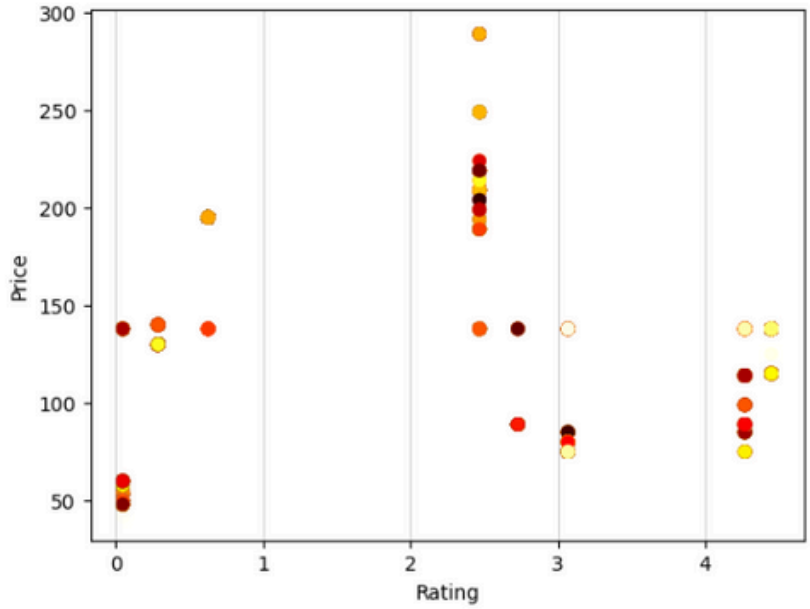
```
M1 = pd.merge(data, df2, on='id', how='inner')
selected_columns = M1[['listing_id', 'reviews_per_month']]

M2 = pd.merge(selected_columns, df3, on='listing_id', how='inner')
output_columns = M2[['reviews_per_month', 'price']]
print(output_columns)
```

```
      reviews_per_month  price
0                  0.29  130.0
1                  0.29  130.0
2                  0.29  130.0
3                  0.29  130.0
4                  0.29  140.0
...                 ...    ...
3280               0.63  195.0
3281               0.63  195.0
3282               0.63  195.0
3283               0.63  195.0
3284               0.63  195.0
```

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

color = np.random.randint(10,100,3285)
plt.scatter(M2['reviews_per_month'], M2['price'], c=color, cmap='hot')
plt.grid(axis='x', alpha=0.5)
plt.xlabel('Rating')
plt.ylabel('Price')
plt.show()
```



Which listings receive the most reviews, and what are their price trends over time?

```
Max_review = max(Data['reviews_per_month'])
# print(Max_review)

M3 = pd.merge(Data, data2, on='id', how='inner')
selected_columns = M3[['listing_id', 'reviews_per_month']]

max_review_row = selected_columns[selected_columns['reviews_per_month'] == Max_review]

M4 = pd.merge(max_review_row[['listing_id']], df3, on='listing_id', how='inner')
print(M4)
```

```
Empty DataFrame
Columns: [listing_id, date, available, price]
Index: []
```

What is the availability of listings with high or low ratings?

```
M5 = pd.merge(Data, data2, on='id', how='inner')
selected_columns = M5[['listing_id', 'reviews_per_month']]
# print(selected_columns)

Max_review = max(selected_columns['reviews_per_month'])
# print(Max_review)
Min_review = min(selected_columns['reviews_per_month'])
# print(Min_review)

max_review_row = selected_columns[selected_columns['reviews_per_month'] == Max_review]
min_review_row = selected_columns[selected_columns['reviews_per_month'] == Min_review]

M6 = pd.merge(max_review_row, df3[['listing_id', 'price']], on='listing_id', how='inner')
M7 = pd.merge(min_review_row, df3[['listing_id', 'price']], on='listing_id', how='inner')

result = pd.concat([M6,M7])
print(result)
```

```
     listing_id  reviews_per_month       price
0        619366               4.45  115.000000
1        619366               4.45  115.000000
2        619366               4.45  115.000000
3        619366               4.45  115.000000
4        619366               4.45  137.944859
..          ...                ...         ...
360        5682               0.05   48.000000
361        5682               0.05   50.000000
362        5682               0.05   50.000000
363        5682               0.05   48.000000
364        5682               0.05  137.944859

[730 rows x 3 columns]
```

Find all listings that have never been reviewed.

```
M8 = pd.merge(Data, data2, on='id', how='right')
selected_columns = M8[['id', 'listing_id']]
print(selected_columns)
```

```
             id  listing_id
0      38917982     7202016
1      39087409     7202016
2      39820030     7202016
3      40813543     7202016
4      41986501     7202016
...         ...         ...
84844  50436321     3624990
84845  51024875     3624990
84846  51511988     3624990
84847  52814482     3624990
84848  56429621     9727246

[84849 rows x 2 columns]
```

Which listings have the least reviews?

```
M8 = pd.merge(Data, data2, on='id', how='inner')
grouped = M8.groupby('listing_id').size().reset_index(name='count')
max_listing_id = grouped.loc[grouped['count'].idxmin()]
print(max_listing_id)
```

```
listing_id    5682
count            1
Name: 0, dtype: int64
```