```
%%capture
!pip install numpy pandas streamlit gdown pyarrow


import os
import shutil

import gdown
import numpy as np
import pandas as pd

# Download files from Google Drive
# Based on data from: http://insideairbnb.com/get-the-data/
file_id_1 = "1m185vTdh-u7_A2ZElBvUD4SCO6oETll2"
file_id_2 = "1w41V1oWHJrBdaNJJQ4oxVBuml5CO7MQX"
downloaded_file_1 = "listings_project.pkl"
downloaded_file_2 = "calendar_project.parquet"
# Download the files from Google Drive
gdown.download(id=file_id_1, output=downloaded_file_1)
gdown.download(id=file_id_2, output=downloaded_file_2)
```

```
Downloading...
From: https://drive.google.com/uc?id=1m185vTdh-u7_A2ZElBvUD4SCO6oETll2
To: /content/listings_project.pkl
100%|██████████| 1.42M/1.42M [00:00<00:00, 134MB/s]
Downloading...
From: https://drive.google.com/uc?id=1w41V1oWHJrBdaNJJQ4oxVBuml5CO7MQX
To: /content/calendar_project.parquet
100%|██████████| 1.23M/1.23M [00:00<00:00, 108MB/s]
'calendar_project.parquet'
```

```
# Show all columns (instead of cascading columns in the middle)
pd.set_option("display.max_columns", None)
# Don't show numbers in scientific notation
pd.set_option("display.float_format", "{:.2f}".format)


df_list = pd.read_pickle("listings_project.pkl")
df_cal = pd.read_parquet("calendar_project.parquet", engine="pyarrow")


df_list.info(verbose=True, show_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6165 entries, 0 to 6172
Data columns (total 34 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   id                                    6165 non-null   int64
 1   host_acceptance_rate                  5365 non-null   float64
 2   host_is_superhost                     6165 non-null   object
 3   host_listings_count                   6165 non-null   int64
 4   host_total_listings_count             6165 non-null   int64
 5   neighbourhood_cleansed                6165 non-null   object
 6   latitude                              6165 non-null   float64
 7   longitude                             6165 non-null   float64
 8   room_type                             6165 non-null   object
 9   accommodates                          6165 non-null   int64
 10  bedrooms                              5859 non-null   float64
 11  beds                                  6082 non-null   float64
 12  amenities                             6165 non-null   int64
 13  price                                 6165 non-null   object
 14  minimum_nights                        6165 non-null   int64
 15  maximum_nights                        6165 non-null   int64
 16  has_availability                      6165 non-null   object
 17  availability_30                       6165 non-null   int64
 18  availability_60                       6165 non-null   int64
 19  availability_90                       6165 non-null   int64
 20  availability_365                      6165 non-null   int64
 21  number_of_reviews                     6165 non-null   int64
 22  number_of_reviews_ltm                 6165 non-null   int64
 23  number_of_reviews_l30d                6165 non-null   int64
 24  review_scores_rating                  5581 non-null   float64
 25  instant_bookable                      6165 non-null   object
 26  reviews_per_month                     5581 non-null   float64
 27  price_in_euros                        0 non-null      object
 28  price_per_person                      6165 non-null   object
 29  minimum_price                         6165 non-null   object
 30  discount_per_5_days_booked            6165 non-null   object
 31  discount_per_10_days_booked           6165 non-null   object
 32  discount_per_30_and_more_days_booked  6165 non-null   object
 33  service_cost                          6165 non-null   object
dtypes: float64(7), int64(14), object(13)
memory usage: 1.6+ MB
```

```
df_list.discount_per_5_days_booked.head(5)
```

```
0    5%
1    5%
2    7%
3    6%
```

```
4    9%
Name: discount_per_5_days_booked, dtype: object
```

```
df_list["discount_per_5_days_booked"] = (df_list["discount_per_5_days_booked"].str.replace("%", "", regex=True).astype("float")* 0.01)
df_list["discount_per_10_days_booked"] = (df_list["discount_per_10_days_booked"].str.replace("%", "", regex=True).astype("float")* 0.01)
df_list["discount_per_30_and_more_days_booked"] = (df_list["discount_per_30_and_more_days_booked"].str.replace("%", "", regex=True).astype(

df_list.discount_per_5_days_booked.head(5)
```

```
0   0.05
1   0.05
2   0.07
3   0.06
4   0.09
Name: discount_per_5_days_booked, dtype: float64
```

```
df_list[["host_is_superhost", "instant_bookable", "has_availability"]].head(5)
```

| | host_is_superhost | instant_bookable | has_availability |
|---|---|---|---|
| **0** | f | t | t |
| **1** | t | f | t |
| **2** | f | f | t |
| **3** | f | f | t |
| **4** | t | f | t |

```
df_list["host_is_superhost"] = (df_list["host_is_superhost"].replace({"f": False, "t": True}).astype("bool"))
df_list["instant_bookable"] = (df_list["instant_bookable"].replace({"f": False, "t": True}).astype("bool"))
df_list["has_availability"] = (df_list["has_availability"].replace({"f": False, "t": True}).astype("bool"))

df_list[["host_is_superhost", "instant_bookable", "has_availability"]].head(5)
```

| | host_is_superhost | instant_bookable | has_availability |
|---|---|---|---|
| **0** | False | True | True |
| **1** | True | False | True |
| **2** | False | False | True |
| **3** | False | False | True |
| **4** | True | False | True |

```
df_list[["price", "price_per_person", "minimum_price", 'service_cost']].head(5)
```

| | price | price_per_person | minimum_price | service_cost |
|---|---|---|---|---|
| **0** | $88.00 | $44 | $176 | $4.99 |
| **1** | $105.00 | $52.5 | $315 | $4.99 |
| **2** | $152.00 | $38 | $304 | $4.99 |
| **3** | $87.00 | $43.5 | $174 | $4.99 |
| **4** | $160.00 | $40 | $320 | $4.99 |

```
df_list["price"] = (df_list["price"].str.replace("$", "", regex=True).str.replace(",", "", regex=True).astype("float"))
df_list["price_per_person"] = (df_list["price_per_person"].str.replace("$", "", regex=True).str.replace(",", "", regex=True).astype("float"
df_list["minimum_price"] = (df_list["minimum_price"].str.replace("$", "", regex=True).str.replace(",", "", regex=True).astype("float"))
df_list["service_cost"] = (df_list["service_cost"].str.replace("$", "", regex=True).str.replace(",", "", regex=True).astype("float"))

df_list[["price", "price_per_person", "minimum_price", 'service_cost']].head(5)
```

| | price | price_per_person | minimum_price | service_cost |
|---|---|---|---|---|
| **0** | 88.00 | 44.00 | 176.00 | 4.99 |
| **1** | 105.00 | 52.50 | 315.00 | 4.99 |
| **2** | 152.00 | 38.00 | 304.00 | 4.99 |
| **3** | 87.00 | 43.50 | 174.00 | 4.99 |
| **4** | 160.00 | 40.00 | 320.00 | 4.99 |

```
df_list = df_list.rename(columns={"price": "price_in_dollar", "neighbourhood_cleansed": "neighbourhood"})
df_list.head()
```

| | id | host_acceptance_rate | host_is_superhost | host_listings_count | host_total_listings_count | neighbourhood | latitu |
|---|---|---|---|---|---|---|---|
| **0** | 23726706 | 0.95 | False | 1 | 1 | IJburg - Zeeburgereiland | 52 |
| **1** | 35815036 | 1.00 | True | 1 | 1 | Noord-Oost | 52 |
| **2** | 31553121 | 1.00 | False | 1 | 1 | Noord-West | 52 |
| **3** | 34745823 | 0.94 | False | 3 | 3 | Gaasperdam - Driemond | 52 |
| | | | | | | Gaasperdam - | |

```
df_list = df_list.astype({"neighbourhood": "category","room_type": "category",})
df_list.head()
```

| | id | host_acceptance_rate | host_is_superhost | host_listings_count | host_total_listings_count | neighbourhood | latitu |
|---|---|---|---|---|---|---|---|
| **0** | 23726706 | 0.95 | False | 1 | 1 | IJburg - Zeeburgereiland | 52 |
| **1** | 35815036 | 1.00 | True | 1 | 1 | Noord-Oost | 52 |
| **2** | 31553121 | 1.00 | False | 1 | 1 | Noord-West | 52 |
| **3** | 34745823 | 0.94 | False | 3 | 3 | Gaasperdam - Driemond | 52 |
| **4** | 44586947 | 0.88 | True | 0 | 0 | Gaasperdam - Driemond | 52 |

```
df_list = df_list.drop(columns=[
        "host_listings_count",
        "host_total_listings_count",
        "availability_60",
        "availability_90",
        "availability_365",
        "number_of_reviews",
        "number_of_reviews_ltm",
        "reviews_per_month",
    ]
)
```

```
df_list.info(verbose=True, show_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6165 entries, 0 to 6172
Data columns (total 26 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   id                                    6165 non-null   int64
 1   host_acceptance_rate                  5365 non-null   float64
 2   host_is_superhost                     6165 non-null   bool
 3   neighbourhood                         6165 non-null   category
 4   latitude                              6165 non-null   float64
 5   longitude                             6165 non-null   float64
 6   room_type                             6165 non-null   category
 7   accommodates                          6165 non-null   int64
 8   bedrooms                              5859 non-null   float64
 9   beds                                  6082 non-null   float64
 10  amenities                             6165 non-null   int64
 11  price_in_dollar                       6165 non-null   float64
 12  minimum_nights                        6165 non-null   int64
 13  maximum_nights                        6165 non-null   int64
 14  has_availability                      6165 non-null   bool
 15  availability_30                       6165 non-null   int64
 16  number_of_reviews_l30d                6165 non-null   int64
 17  review_scores_rating                  5581 non-null   float64
 18  instant_bookable                      6165 non-null   bool
 19  price_in_euros                        0 non-null      object
 20  price_per_person                      6165 non-null   float64
 21  minimum_price                         6165 non-null   float64
 22  discount_per_5_days_booked            6165 non-null   float64
 23  discount_per_10_days_booked           6165 non-null   float64
 24  discount_per_30_and_more_days_booked  6165 non-null   float64
 25  service_cost                          6165 non-null   float64
dtypes: bool(3), category(2), float64(13), int64(7), object(1)
memory usage: 1.1+ MB
```

```
df_list["price_in_euros"].unique
```

```
<bound method Series.unique of 0       None
1       None
```

```
2       None
3       None
4       None
        ...
6168    None
6169    None
6170    None
6171    None
6172    None
Name: price_in_euros, Length: 6165, dtype: object>
```

```
df_list = df_list.drop(columns=["price_in_euros"])
```

```
df_list = df_list.dropna(subset=["review_scores_rating", "host_acceptance_rate"])
```

```
df_list["room_type"].unique()
```

```
['Private room', 'Entire home/apt', 'Hotel room', 'Shared room']
Categories (4, object): ['Entire home/apt', 'Hotel room', 'Private room', 'Shared room']
```

```
def fill_empty_bedrooms(accommodates: int, bedrooms: int, room_type: str) -> int:
    if (room_type == "Private room") or (room_type == "Shared room"):
        return 1
    elif (room_type == "Hotel room") or (room_type == "Entire home/apt"):
        return np.ceil(accommodates / 2)
    else:
        return bedrooms
```

```
%%timeit -r 4 -n 100

temp_df = df_list.copy()  # Deep copy of the df, not a "view"
temp_df["rooms"] = df_list[["accommodates", "bedrooms", "room_type"]].apply(
    lambda x: fill_empty_bedrooms(x["accommodates"], x["bedrooms"], x["room_type"]),
    axis=1,
)
```

```
92.6 ms ± 31.9 ms per loop (mean ± std. dev. of 4 runs, 100 loops each)
```

```
df_list["bedrooms"] = df_list[["accommodates", "bedrooms", "room_type"]].apply(
    lambda x: fill_empty_bedrooms(x["accommodates"], x["bedrooms"], x["room_type"]),
    axis=1,
)
```

**Related functions (In *general* order of preference)**

- apply(): Apply a function along one or multiple columns
- pipe(): Chain multiple transformations/functions after each other
- applymap(): Use strictly as a transformation of current value to a new value
- itertuples(): Iterate over DataFrame rows as named tuples
- iteritems(): Iterate over DataFrame columns
- iterrows(): Iterate over DataFrame rows

```
%%timeit -r 4 -n 100

temp_df = df_list.copy()

# Please use as many lines as you think you need to
# implement this function. We required 5 separate
# statements.
temp_df["beds"] = temp_df.bedrooms
priv_shared_mask = (temp_df.room_type == "Private room") | (
    temp_df.room_type == "Shared room"
)
temp_df.loc[priv_shared_mask, "beds"] = 1
hotel_apt_mask = (temp_df.room_type == "Hotel room") | (
    temp_df.room_type == "Entire home/apt"
)
temp_df.loc[hotel_apt_mask, "beds"] = np.ceil(temp_df.accommodates / 2)
```

```
3.05 ms ± 142 µs per loop (mean ± std. dev. of 4 runs, 100 loops each)
```

```
df_list = df_list.dropna(subset=["bedrooms", "beds"])
```

```
df_list["beds"] = df_list["beds"].astype("int")
df_list["bedrooms"] = df_list["bedrooms"].astype("int")
```

```
<ipython-input-25-b8100156c6b9>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
  df_list["beds"] = df_list["beds"].astype("int")
<ipython-input-25-b8100156c6b9>:2: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
  df_list["bedrooms"] = df_list["bedrooms"].astype("int")
```

```
df_list.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4817 entries, 0 to 6172
Data columns (total 25 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   id                                    4817 non-null   int64
 1   host_acceptance_rate                  4817 non-null   float64
 2   host_is_superhost                     4817 non-null   bool
 3   neighbourhood                         4817 non-null   category
 4   latitude                              4817 non-null   float64
 5   longitude                             4817 non-null   float64
 6   room_type                             4817 non-null   category
 7   accommodates                          4817 non-null   int64
 8   bedrooms                              4817 non-null   int64
 9   beds                                  4817 non-null   int64
 10  amenities                             4817 non-null   int64
 11  price_in_dollar                       4817 non-null   float64
 12  minimum_nights                        4817 non-null   int64
 13  maximum_nights                        4817 non-null   int64
 14  has_availability                      4817 non-null   bool
 15  availability_30                       4817 non-null   int64
 16  number_of_reviews_l30d                4817 non-null   int64
 17  review_scores_rating                  4817 non-null   float64
 18  instant_bookable                      4817 non-null   bool
 19  price_per_person                      4817 non-null   float64
 20  minimum_price                         4817 non-null   float64
 21  discount_per_5_days_booked            4817 non-null   float64
 22  discount_per_10_days_booked           4817 non-null   float64
 23  discount_per_30_and_more_days_booked  4817 non-null   float64
 24  service_cost                          4817 non-null   float64
dtypes: bool(3), category(2), float64(11), int64(9)
memory usage: 814.7 KB
```

```
df_list["accommodates"] = df_list["accommodates"].astype('int8')
df_list["bedrooms"] = df_list["bedrooms"].astype('int8')
df_list["beds"] = df_list["beds"].astype('int8')
df_list["amenities"] = df_list["amenities"].astype('int8')
df_list["minimum_nights"] = df_list["minimum_nights"].astype('int8')
df_list["maximum_nights"] = df_list["maximum_nights"].astype('int8')
df_list["availability_30"] = df_list["availability_30"].astype('int8')
df_list["number_of_reviews_l30d"] = df_list["number_of_reviews_l30d"].astype('int8')
```

```
<ipython-input-27-7d32cd0772bb>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
  df_list["accommodates"] = df_list["accommodates"].astype('int8')
<ipython-input-27-7d32cd0772bb>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
  df_list["bedrooms"] = df_list["bedrooms"].astype('int8')
<ipython-input-27-7d32cd0772bb>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
  df_list["beds"] = df_list["beds"].astype('int8')
<ipython-input-27-7d32cd0772bb>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
  df_list["amenities"] = df_list["amenities"].astype('int8')
<ipython-input-27-7d32cd0772bb>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
  df_list["minimum_nights"] = df_list["minimum_nights"].astype('int8')
<ipython-input-27-7d32cd0772bb>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
  df_list["maximum_nights"] = df_list["maximum_nights"].astype('int8')
<ipython-input-27-7d32cd0772bb>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
  df_list["availability_30"] = df_list["availability_30"].astype('int8')
<ipython-input-27-7d32cd0772bb>:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
    Try using .loc[row_indexer,col_indexer] = value instead

    See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
      df_list["number_of_reviews_l30d"] = df_list["number_of_reviews_l30d"].astype('int8')


df_list["host_acceptance_rate"] = df_list["host_acceptance_rate"].astype('float16')
df_list["latitude"] = df_list["latitude"].astype('float16')
df_list["longitude"] = df_list["longitude"].astype('float16')
df_list["review_scores_rating"] = df_list["review_scores_rating"].astype('float16')
df_list["price_in_dollar"] = df_list["price_in_dollar"].astype('float16')
df_list["price_per_person"] = df_list["price_per_person"].astype('float16')
df_list["minimum_price"] = df_list["minimum_price"].astype('float16')
df_list["discount_per_5_days_booked"] = df_list["discount_per_5_days_booked"].astype('float16')
df_list["discount_per_10_days_booked"] = df_list["discount_per_10_days_booked"].astype('float16')
df_list["discount_per_30_and_more_days_booked"] = df_list["discount_per_30_and_more_days_booked"].astype('float16')
df_list["service_cost"] = df_list["service_cost"].astype('float16')


    Try using .loc[row_indexer,col_indexer] = value instead

    See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
      df_list["latitude"] = df_list["latitude"].astype('float16')
    <ipython-input-28-ec85826f01e4>:3: SettingWithCopyWarning:
    A value is trying to be set on a copy of a slice from a DataFrame.
    Try using .loc[row_indexer,col_indexer] = value instead

    See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
      df_list["longitude"] = df_list["longitude"].astype('float16')
    <ipython-input-28-ec85826f01e4>:4: SettingWithCopyWarning:
    A value is trying to be set on a copy of a slice from a DataFrame.
    Try using .loc[row_indexer,col_indexer] = value instead

    See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
      df_list["review_scores_rating"] = df_list["review_scores_rating"].astype('float16')
    <ipython-input-28-ec85826f01e4>:5: SettingWithCopyWarning:
    A value is trying to be set on a copy of a slice from a DataFrame.
    Try using .loc[row_indexer,col_indexer] = value instead

    See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
      df_list["price_in_dollar"] = df_list["price_in_dollar"].astype('float16')
    <ipython-input-28-ec85826f01e4>:6: SettingWithCopyWarning:
    A value is trying to be set on a copy of a slice from a DataFrame.
    Try using .loc[row_indexer,col_indexer] = value instead

    See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
      df_list["price_per_person"] = df_list["price_per_person"].astype('float16')
    <ipython-input-28-ec85826f01e4>:7: SettingWithCopyWarning:
    A value is trying to be set on a copy of a slice from a DataFrame.
    Try using .loc[row_indexer,col_indexer] = value instead

    See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
      df_list["minimum_price"] = df_list["minimum_price"].astype('float16')
    <ipython-input-28-ec85826f01e4>:8: SettingWithCopyWarning:
    A value is trying to be set on a copy of a slice from a DataFrame.
    Try using .loc[row_indexer,col_indexer] = value instead

    See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
      df_list["discount_per_5_days_booked"] = df_list["discount_per_5_days_booked"].astype('float16')
    <ipython-input-28-ec85826f01e4>:9: SettingWithCopyWarning:
    A value is trying to be set on a copy of a slice from a DataFrame.
    Try using .loc[row_indexer,col_indexer] = value instead

    See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
      df_list["discount_per_10_days_booked"] = df_list["discount_per_10_days_booked"].astype('float16')
    <ipython-input-28-ec85826f01e4>:10: SettingWithCopyWarning:
    A value is trying to be set on a copy of a slice from a DataFrame.
    Try using .loc[row_indexer,col_indexer] = value instead

    See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
      df_list["discount_per_30_and_more_days_booked"] = df_list["discount_per_30_and_more_days_booked"].astype('float16')
    <ipython-input-28-ec85826f01e4>:11: SettingWithCopyWarning:
    A value is trying to be set on a copy of a slice from a DataFrame.
    Try using .loc[row_indexer,col_indexer] = value instead

    See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
      df_list["service_cost"] = df_list["service_cost"].astype('float16')


df_list.info()

    <class 'pandas.core.frame.DataFrame'>
    Int64Index: 4817 entries, 0 to 6172
    Data columns (total 25 columns):
     #   Column                                Non-Null Count  Dtype
    ---  ------                                --------------  -----
     0   id                                    4817 non-null   int64
     1   host_acceptance_rate                  4817 non-null   float16
     2   host_is_superhost                     4817 non-null   bool
     3   neighbourhood                         4817 non-null   category
     4   latitude                              4817 non-null   float16
     5   longitude                             4817 non-null   float16
     6   room_type                             4817 non-null   category
     7   accommodates                          4817 non-null   int8
     8   bedrooms                              4817 non-null   int8
     9   beds                                  4817 non-null   int8
```

```
 10  amenities                             4817 non-null   int8
 11  price_in_dollar                       4817 non-null   float16
 12  minimum_nights                        4817 non-null   int8
 13  maximum_nights                        4817 non-null   int8
 14  has_availability                      4817 non-null   bool
 15  availability_30                       4817 non-null   int8
 16  number_of_reviews_l30d                4817 non-null   int8
 17  review_scores_rating                  4817 non-null   float16
 18  instant_bookable                      4817 non-null   bool
 19  price_per_person                      4817 non-null   float16
 20  minimum_price                         4817 non-null   float16
 21  discount_per_5_days_booked            4817 non-null   float16
 22  discount_per_10_days_booked           4817 non-null   float16
 23  discount_per_30_and_more_days_booked  4817 non-null   float16
 24  service_cost                          4817 non-null   float16
dtypes: bool(3), category(2), float16(11), int64(1), int8(8)
memory usage: 240.8 KB
```

```
%%timeit -r 4 -n 100

temp_df = df_list.copy()  # Deep copy of the df, not a "view"
temp_df["rooms"] = df_list[["accommodates", "bedrooms", "room_type"]].apply(
    lambda x: fill_empty_bedrooms(x["accommodates"], x["bedrooms"], x["room_type"]),
    axis=1,
)
```

```
64.9 ms ± 7.24 ms per loop (mean ± std. dev. of 4 runs, 100 loops each)
```

```
df_list.info(verbose=True, show_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4817 entries, 0 to 6172
Data columns (total 25 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   id                                    4817 non-null   int64
 1   host_acceptance_rate                  4817 non-null   float16
 2   host_is_superhost                     4817 non-null   bool
 3   neighbourhood                         4817 non-null   category
 4   latitude                              4817 non-null   float16
 5   longitude                             4817 non-null   float16
 6   room_type                             4817 non-null   category
 7   accommodates                          4817 non-null   int8
 8   bedrooms                              4817 non-null   int8
 9   beds                                  4817 non-null   int8
 10  amenities                             4817 non-null   int8
 11  price_in_dollar                       4817 non-null   float16
 12  minimum_nights                        4817 non-null   int8
 13  maximum_nights                        4817 non-null   int8
 14  has_availability                      4817 non-null   bool
 15  availability_30                       4817 non-null   int8
 16  number_of_reviews_l30d                4817 non-null   int8
 17  review_scores_rating                  4817 non-null   float16
 18  instant_bookable                      4817 non-null   bool
 19  price_per_person                      4817 non-null   float16
 20  minimum_price                         4817 non-null   float16
 21  discount_per_5_days_booked            4817 non-null   float16
 22  discount_per_10_days_booked           4817 non-null   float16
 23  discount_per_30_and_more_days_booked  4817 non-null   float16
 24  service_cost                          4817 non-null   float16
dtypes: bool(3), category(2), float16(11), int64(1), int8(8)
memory usage: 240.8 KB
```

```
df_list.head(3)
```

| | id | host_acceptance_rate | host_is_superhost | neighbourhood | latitude | longitude | room_type | accommodates | bedrooms |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 23726706 | 0.95 | False | IJburg - Zeeburgereiland | 52.34 | 4.98 | Private room | 2 | 1 |
| **1** | 35815036 | 1.00 | True | Noord-Oost | 52.44 | 4.96 | Entire home/apt | 2 | 1 |
| **2** | 31553121 | 1.00 | False | Noord-West | 52.44 | 4.92 | Entire home/apt | 4 | 2 |

```
# The Calendar DataFrame!
df_cal.head(3)
```

| | listing_id | date | available | price_in_dollar | minimum_nights | maximum_nights |
|---|---|---|---|---|---|---|
| **0** | 23726706 | 2022-06-05 | False | 90.00 | 2 | 1125 |
| **1** | 23726706 | 2022-06-06 | False | 90.00 | 2 | 1125 |
| **2** | 23726706 | 2022-06-07 | False | 90.00 | 2 | 1125 |

```
# First start by making a copy, for debugging purposes
calendar_newdf = df_cal.copy()

include_list = (calendar_newdf["minimum_nights"] >= 3)


# Get all the listings with a minimum nights of 3+
# Use the include_list
calendar_newdf = calendar_newdf.loc[include_list]
```

**Related functions**

- isin(): Filter the DataFrame on provided values
- eq(): Filter the DataFrame for all values equal to the provided input
- ne(): Filter the DataFrame for all values not equal to the provided input

```
calendar_newdf["five_day_dollar_price"] = calendar_newdf["price_in_dollar"] * 5
```

```
<ipython-input-36-e85efa2d7a46>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
  calendar_newdf["five_day_dollar_price"] = calendar_newdf["price_in_dollar"] * 5
```

Now let's transform our newly created DataFrame into a **pivot table**, where we aggregate our rows using the `listing_id` as the index, and the columns `available` and `five_day_dollar_price` as values.

```
calendar_summarizeddf = pd.pivot_table(
    data=calendar_newdf,
    index=["listing_id"],
    values=["available", "five_day_dollar_price"],
    aggfunc=np.mean,  # The default aggregation function used
    # for merging multiple related rows of data.
)

calendar_summarizeddf.head(3)
```

| listing_id | available | five_day_dollar_price |
|---|---|---|
| **2818** | 0.21 | 346.90 |
| **44391** | 0.00 | 1200.00 |
| **49552** | 0.46 | 1162.50 |

```
temp_sum_df = pd.pivot_table(
    data=calendar_newdf,
    index=["listing_id"],
    values=["price_in_dollar"],
    aggfunc=np.max,  # The default aggregation function used
    # for merging multiple related rows of data.
)

temp_sum_df.head(3)
```

| listing_id | price_in_dollar |
|---|---|
| **2818** | 80.00 |
| **44391** | 240.00 |
| **49552** | 300.00 |

```
final_df = pd.merge(
    df_list,
    calendar_summarizeddf,
    left_on=["id"],
    right_on=["listing_id"],
    how="left",
)
final_df.head(3)
```

| | id | host_acceptance_rate | host_is_superhost | neighbourhood | latitude | longitude | room_type | accommodates | bedrooms |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 23726706 | 0.95 | False | IJburg - | 52.34 | 4.98 | Private | 2 | 1 |

```
# Merging dataframes
final_df = df_list.set_index("id").join(calendar_summarizeddf, how="left")

final_df.head(3)
```

| id | host_acceptance_rate | host_is_superhost | neighbourhood | latitude | longitude | room_type | accommodates | bedrooms | b |
|---|---|---|---|---|---|---|---|---|---|
| **23726706** | 0.95 | False | IJburg - Zeeburgereiland | 52.34 | 4.98 | Private room | 2 | 1 | |
| **35815036** | 1.00 | True | Noord-Oost | 52.44 | 4.96 | Entire home/apt | 2 | 1 | |
| **31553121** | 1.00 | False | Noord-West | 52.44 | 4.92 | Entire home/apt | 4 | 2 | |

```
final_df.groupby(by=["room_type"])[["review_scores_rating","five_day_dollar_price",]].median()
```

| room_type | review_scores_rating | five_day_dollar_price |
|---|---|---|
| **Entire home/apt** | 4.88 | 975.00 |
| **Hotel room** | 4.75 | 1110.16 |
| **Private room** | 4.78 | 710.91 |
| **Shared room** | 4.57 | 724.11 |

```
final_df.to_csv(
    "WK2_Airbnb_Amsterdam_listings_proj_solution.csv",
    index=True,
)


from google.colab import files

# Download the file locally
files.download('WK2_Airbnb_Amsterdam_listings_proj_solution.csv')



%%writefile app.py
import pandas as pd
import streamlit as st
from pandas.api.types import (
    is_categorical_dtype,
    is_datetime64_any_dtype,
    is_numeric_dtype,
    is_object_dtype
)

st.title("Filter your Airbnb Listings dataframe!")

st.write(
    """This app is based on this blog [here](https://blog.streamlit.io/auto-generate-a-dataframe-filtering-ui-in-streamlit-with-filter_data
    Can you think of ways to extend it with visuals?
    """
)


def filter_dataframe(df: pd.DataFrame) -> pd.DataFrame:
    """
    Adds a UI on top of a dataframe to let viewers filter columns
    Args:
        df (pd.DataFrame): Original dataframe
    Returns:
        pd.DataFrame: Filtered dataframe
    """
    modify = st.checkbox("Add filters")

    if not modify:
        return df

    df = df.copy()

    # Try to convert datetimes into a standard format (datetime, no timezone)
    for col in df.columns:
```

```
        if is_object_dtype(df[col]):
            try:
                df[col] = pd.to_datetime(df[col])
            except Exception:
                pass

        if is_datetime64_any_dtype(df[col]):
            df[col] = df[col].dt.tz_localize(None)

    modification_container = st.container()

    with modification_container:
        to_filter_columns = st.multiselect("Filter dataframe on", df.columns)
        for column in to_filter_columns:
            left, right = st.columns((1, 20))
            left.write("↳")
            # Treat columns with < 10 unique values as categorical
            if is_categorical_dtype(df[column]) or df[column].nunique() < 10:
                user_cat_input = right.multiselect(
                    f"Values for {column}",
                    df[column].unique(),
                    default=list(df[column].unique()),
                )
                df = df[df[column].isin(user_cat_input)]
            elif is_numeric_dtype(df[column]):
                _min = float(df[column].min())
                _max = float(df[column].max())
                step = (_max - _min) / 100
                user_num_input = right.slider(
                    f"Values for {column}",
                    _min,
                    _max,
                    (_min, _max),
                    step=step,
                )
                df = df[df[column].between(*user_num_input)]
            elif is_datetime64_any_dtype(df[column]):
                user_date_input = right.date_input(
                    f"Values for {column}",
                    value=(
                        df[column].min(),
                        df[column].max(),
                    ),
                )
                if len(user_date_input) == 2:
                    user_date_input = tuple(map(pd.to_datetime, user_date_input))
                    start_date, end_date = user_date_input
                    df = df.loc[df[column].between(start_date, end_date)]
            else:
                user_text_input = right.text_input(
                    f"Substring or regex in {column}",
                )
                if user_text_input:
                    df = df[df[column].str.contains(user_text_input)]

    return df


df = pd.read_csv(
    "WK2_Airbnb_Amsterdam_listings_proj_solution.csv", index_col=0
)
st.dataframe(filter_dataframe(df))
```

```
Writing app.py
```

The **%%writefile [FILE_NAME].[FILE_EXTENSION]** command let's us save the code written in the cells in your Google Colab instance. Having it saved like that enables us to download it as a file, as seen below.

```
from google.colab import files

# Download the file locally
files.download('app.py')
```

```
%%writefile requirements.txt
pandas
streamlit
```

```
Writing requirements.txt
```

```
from google.colab import files

# Download the file locally
files.download('requirements.txt')
```

✓ 0s completed at 16:07