Concat-Merge

October 7, 2024

1 Concat and Merge

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

1.1 Concat example

4 months of orders

[2]:
```
jan_orders = pd.read_csv("https://raw.githubusercontent.com/mafudge/datasets/
↪master/delimited/jan-orders.csv")
jan_orders
```

[2]:
```
   order_id  order_date  order_amount
0      1023  2024-01-05          56.9
1      1024  2024-01-17         146.7
2      1025  2024-01-25          36.4
```

[3]:
```
feb_orders = pd.read_csv("https://raw.githubusercontent.com/mafudge/datasets/
↪master/delimited/feb-orders.csv")
feb_orders
```

[3]:
```
   order_id  order_date  order_amount
0      1026  2024-02-10        104.35
1      1027  2024-02-24         33.70
```

The code below addresses the following…

- Create a list of months: months = [“jan”, “feb”, “mar”, “apr”] defines the list of months you’re interested in.
- Initialize an empty list orders: This will store DataFrames for each month’s order data.
- Loop through each month:
  - The loop reads a CSV file for each month from a given URL using pd.read_csv().
  - The resulting DataFrame order is appended to the orders list.
- Concatenate all DataFrames:

– pd.concat(orders, ignore_index=True) concatenates the list of DataFrames (orders) into a single DataFrame df.
– The ignore_index=True argument ensures resulting DataFrame has a continuous index (starting from 0) without keeping the original indices from individual DataFrames.
– Finally, the concatenated DataFrame df will contain the data from all four months.

[4]:
```
months = ["jan", "feb", "mar", "apr"]
orders = [] #list for order dataframes
for month in months:
    order = pd.read_csv(f"https://raw.githubusercontent.com/mafudge/datasets/
 ↪master/delimited/{month}-orders.csv")
    orders.append(order) # append order dataframe to list

# make one dataframe from the list of dataframes
df = pd.concat(orders, ignore_index=True)
df
```

[4]:
```
    order_id  order_date  order_amount
0       1023  2024-01-05         56.90
1       1024  2024-01-17        146.70
2       1025  2024-01-25         36.40
3       1026  2024-02-10        104.35
4       1027  2024-02-24         33.70
5       1028  2024-03-06         86.50
6       1029  2024-03-22        209.00
7       1030  2024-03-30        136.55
8       1031  2024-04-01        256.00
9       1032  2024-04-09         42.30
10      1033  2024-04-17        199.20
11      1034  2024-04-29         26.88
```

1.2 Merge example

We have 2 data files

[13]:
```
players = pd.read_csv("https://raw.githubusercontent.com/mafudge/datasets/
 ↪master/delimited/bbplayers.csv")
players
```

[13]:
```
   player_id player_name  career_pts  player_team_id
0        101      Jordan       32292             1.0
1        102      Pippen       18940             1.0
2        103      Bryant       33643             2.0
3        104      O'Neal       28596             2.0
4        105       Fudge           0             NaN
```

[14]:
```
teams = pd.read_csv("https://raw.githubusercontent.com/mafudge/datasets/master/
 ↪delimited/bbteams.csv")
```

```
teams
```

```
[14]:    team_id team_name    team_location
      0        1     Bulls      Chicago, IL
      1        2    Lakers  Los Angeles, CA
      2        3   Tropics        Flint, MI
```

The code performs an inner join between two DataFrames, players and teams, based on common columns:

- left=players: This specifies the left DataFrame for the merge.
- right=teams: This specifies the right DataFrame for the merge.
- how="inner": Indicates you want an inner join. This means only rows with matching values in both DataFrames will be included in the final DataFrame.
- left_on="player_team_id": This specifies the column in the players DataFrame to join on.
- right_on="team_id": This specifies the column in the teams DataFrame to join on.

The resulting DataFrame playersteams will contain only the rows where there is a match between player_team_id in the players DataFrame and team_id in the teams DataFrame. This merged DataFrame include columns from both players and teams DataFrames, with rows only where the specified IDs match.

```
[15]: playersteams = pd.merge(
              left=players,
              right=teams,
              how="inner",
              left_on="player_team_id",
              right_on="team_id")
      playersteams
```

```
[15]:    player_id player_name  career_pts  player_team_id  team_id team_name  \
      0        101      Jordan       32292             1.0        1     Bulls
      1        102     Pippen       18940             1.0        1     Bulls
      2        103      Bryant       33643             2.0        2    Lakers
      3        104      O'Neal       28596             2.0        2    Lakers

           team_location
      0      Chicago, IL
      1      Chicago, IL
      2  Los Angeles, CA
      3  Los Angeles, CA
```

1.2.1 Code performs a left join between the players and teams DataFrames:

- left=players: The left DataFrame (players) is the primary DataFrame for the merge.
- right=teams: The right DataFrame (teams) is the secondary DataFrame to merge with.
- how="left": A left join will be done. All rows from players will be included and matching rows from the teams DataFrame will be included. Rows from players that do not have matching

rows in teams will still be included, with NaN values for columns from teams.
- left_on="player_team_id": specifies the column in the players DataFrame to join on.
- right_on="team_id": specifies the column in the teams DataFrame to join on.

The resulting DataFrame allplayers will contain all rows from players and corresponding rows from teams where the IDs match. If there are no matching rows in teams, the resulting columns from teams will contain NaN. This type of join is useful when you want to retain all entries from the primary DataFrame and include additional data from the secondary DataFrame.

[16]:
```
allplayers = pd.merge(
        left=players,
        right=teams,
        how="left",
        left_on="player_team_id",
        right_on="team_id")
allplayers
```

[16]:
```
   player_id player_name  career_pts  player_team_id  team_id team_name  \
0        101      Jordan       32292             1.0      1.0     Bulls
1        102      Pippen       18940             1.0      1.0     Bulls
2        103      Bryant       33643             2.0      2.0    Lakers
3        104      O'Neal       28596             2.0      2.0    Lakers
4        105       Fudge           0             NaN      NaN       NaN

     team_location
0      Chicago, IL
1      Chicago, IL
2  Los Angeles, CA
3  Los Angeles, CA
4              NaN
```

A right join between the players and teams DataFrames:

- left=players: The left DataFrame is players.
- right=teams: The right DataFrame is teams.
- how="right": A right join should be performed. This means all rows from the teams DataFrame will be included in the resulting DataFrame, and matching rows from the players DataFrame will be included where available. Rows from teams that do not have matching rows in players will still be included, with NaN values for columns from players.
- left_on="player_team_id": The column in the players DataFrame to join on.
- right_on="team_id": The column in the teams DataFrame to join on.

The resulting DataFrame allteams contains all rows from teams and the corresponding rows from players where the IDs match. If there are no matching rows in players, the resulting columns from players will contain NaN.

```
[17]: allteams = pd.merge(left=players, right=teams, how="right",␣
       ↪left_on="player_team_id", right_on="team_id")
      allteams
```

```
[17]:    player_id player_name  career_pts  player_team_id  team_id team_name  \
      0      101.0      Jordan     32292.0             1.0        1     Bulls
      1      102.0      Pippen     18940.0             1.0        1     Bulls
      2      103.0      Bryant     33643.0             2.0        2    Lakers
      3      104.0      O'Neal     28596.0             2.0        2    Lakers
      4        NaN         NaN         NaN             NaN        3   Tropics

             team_location
      0        Chicago, IL
      1        Chicago, IL
      2    Los Angeles, CA
      3    Los Angeles, CA
      4          Flint, MI
```

Code performs an outer join between the players and teams DataFrames:

- left=players: The left DataFrame is players.
- right=teams: The right DataFrame is teams.
- how="outer": Specifies an outer join which returns all rows from both DataFrames. If there are matching rows they are combined. If a row in one DataFrame does not have a matching row in the other DataFrame, the missing values will be filled with NaN.
- left_on="player_team_id": The column from the players DataFrame to use for the join.
- right_on="team_id": The column from the teams DataFrame to use for the join.

The resulting DataFrame allplayersteams will include:

- All rows from players and teams.
- Matching rows from both DataFrames where the player_team_id from players matches the team_id from teams.
- For non-matching rows, columns from the missing DataFrame will contain NaN.

Useful when you want to combine data from both DataFrames without losing any rows, regardless of whether a match between the columns.

```
[19]: allplayersteams = pd.merge(left=players, right=teams, how="outer",␣
       ↪left_on="player_team_id", right_on="team_id")
      allplayersteams
```

```
[19]:    player_id player_name  career_pts  player_team_id  team_id team_name  \
      0      101.0      Jordan     32292.0             1.0      1.0     Bulls
      1      102.0      Pippen     18940.0             1.0      1.0     Bulls
      2      103.0      Bryant     33643.0             2.0      2.0    Lakers
      3      104.0      O'Neal     28596.0             2.0      2.0    Lakers
      4      105.0       Fudge         0.0             NaN      NaN       NaN
```

```
5        NaN         NaN         NaN              NaN      3.0   Tropics

     team_location
0      Chicago, IL
1      Chicago, IL
2  Los Angeles, CA
3  Los Angeles, CA
4              NaN
5        Flint, MI
```

```
[ ]:
```