

Using Outer Join to Select Actors - Full Solution

Exercise

Using outer join to select actors

One cool aspect of using an outer join is that, because it returns all rows from both merged tables and null where they do not match, you can use it to find rows that do not have a match in the other table. To try for yourself, you have been given two tables with a list of actors from two popular movies: Iron Man and Iron Man 2. Most of the actors played in both movies. Use an outer join to find actors who did not act in both movies.

The Iron Man table is called `iron_1_actors`, and Iron Man 2 table is called `iron_2_actors`. Both tables have been loaded for you and a few rows printed so you can see the structure.

Instructions

- Save to `iron_1_and_2` the merge of `iron_1_actors` (left) with `iron_2_actors` tables with an outer join on the `id` column, and set suffixes to `('_1', '_2')`.
- Create an index that returns True if `name_1` or `name_2` are null, and False otherwise.

Code Answer:

```
1 # Merge iron_1_actors to iron_2_actors on id with outer join using suffixes
2 iron_1_and_2 = iron_1_actors.merge(iron_2_actors, on='id', how='outer',
3                                     suffixes=('_1', '_2'))
4
5
6
7 # Create an index that returns True if name_1 or name_2 are null
8 m = ((iron_1_and_2['name_1'].isnull() | iron_1_and_2['name_2'].isnull())
9       (iron_1_and_2['_1'].isnull() | iron_1_and_2['_2'].isnull()))
10
11 # Print the first few rows of iron_1_and_2 where the condition is True
12 print(iron_1_and_2[m].head())
```

Python Shell

```
iron_1_actors
  character  id      name
1    Vinson  17857  Shaun Toub
4  Virginia "Popper" Putts  12052  Gwyneth Paltrow

iron_2_actors
  character  id      name
4    Metalic Rushean / Natasha Romanoff / Black Widow  1245  Scarlett Johansson
5    Mickey Rourke  1245  Mickey Rourke
```

Screenshot showing the exercise context for using an outer join to select actors.

Code Answer:

```
# Merge iron_1_actors to iron_2_actors on id with outer join using suffixes
iron_1_and_2 = iron_1_actors.merge(iron_2_actors, on='id', how='outer',
                                   suffixes=('_1', '_2'))
```

```
# Create an index that returns True if name_1 or name_2 are null
m = iron_1_and_2['name_1'].isnull() | iron_1_and_2['name_2'].isnull()
```

```
# Print the first few rows of iron_1_and_2 where the condition is True
print(iron_1_and_2[m].head())
```

Explanation:

1. The `merge` function performs an outer join on the `'id'` column between the `'iron_1_actors'` and `'iron_2_actors'` DataFrames. The `how='outer'` parameter ensures that all rows from both DataFrames are included, with missing values filled with `NaN` where matches are not found. The

``suffixes=('_1', '_2')`` parameter adds suffixes to distinguish columns from the two DataFrames.

2. The condition ``m`` identifies rows where either 'name_1' or 'name_2' is null, using the ``isnull()`` method combined with the logical OR operator (``|``). This step isolates rows corresponding to actors who appear in only one of the two movies.

3. The final step filters the 'iron_1_and_2' DataFrame using the condition ``m`` and displays the first few rows with ``print(iron_1_and_2[m].head())``. This allows verification of actors who are exclusive to one movie.