

## Self Join - Updated Solution with Print Output

The screenshot shows a web browser window with a DataCamp course page. The browser tabs include 'Self join | Python', 'docx maker - Right Join Movie', 'ChatGPT', 'Tool Setup Assistant', and 'Installer/matt: Inst is an in:'. The address bar shows 'campus.datacamp.com/courses/joining-data-with-pandas/merging-tables-with-different-join-types?ex=10'. A warning message at the top states: 'You are using an unsupported command-line flag: --unsafely-treat-insecure-origin-as-secure=http://54.173.176.93:4444. Stability and security will suffer.'

The main content area is titled 'Exercise' and 'Self join'. It contains instructions: 'Merging a table to itself can be useful when you want to compare values in a column to other values in the same column. In this exercise, you will practice this by creating a table that for each movie will list the movie director and a member of the crew on one row. You have been given a table called `crews`, which has columns `id`, `job`, and `name`. First, merge the table to itself using the movie ID. This merge will give you a larger table where for each movie, every job is matched against each other. Then select only those rows with a director in the left table, and avoid having a row where the director's job is listed in both the left and right tables. This filtering will remove job combinations that aren't with the director.'

Below the instructions is a 'script.py' editor with the following code:

```
1 # Merge the crews table to itself
2 crews_self_merged = crews.merge(crews, on='id', how='inner',
3                                   suffixes=('_dir', '_crew'))
4
5 # Create a Boolean index to select the appropriate rows
6 boolean_filter = ((crews_self_merged['job_dir'] == 'Director') &
7                   (crews_self_merged['job_crew'] != 'Director'))
8 direct_crews = crews_self_merged[boolean_filter]
9
10 # Print the first few rows of direct_crews
11 print(_____)
```

At the bottom of the editor are 'Run Code' and 'Submit Answer' buttons. Below the editor is a 'Python Shell' window showing the output of the code.

Screenshot showing the exercise context for performing a self join on the crews table.

### Code Answer:

```
# Merge the crews table to itself
crews_self_merged = crews.merge(crews, on='id', how='inner',
                                suffixes=('_dir', '_crew'))

# Create a Boolean index to select the appropriate rows
boolean_filter = ((crews_self_merged['job_dir'] == 'Director') &
                  (crews_self_merged['job_crew'] != 'Director'))
direct_crews = crews_self_merged[boolean_filter]

# Print the first few rows of direct_crews
print(direct_crews.head())
```

### Explanation:

1. The `merge` function performs a self join on the 'crews' table by joining it to itself using the 'id' column. The `how='inner'` parameter ensures that only rows with matching 'id' values in both tables are included. The

``suffixes=('_dir', '_crew')`` parameter is used to distinguish between columns from the left and right tables, representing directors and crew members respectively.

2. A Boolean filter is created to select rows where the 'job\_dir' column is 'Director' and the 'job\_crew' column is not 'Director'. This ensures that only valid combinations of directors and crew members are included.

3. The resulting DataFrame is filtered using the Boolean index to produce 'direct\_crews', which contains the desired pairs of directors and their corresponding crew members.

4. Finally, the ``head()`` method is used to print the first few rows of the 'direct\_crews' DataFrame, allowing verification of the result.