Types of Pandas Joins and How to use them in Python

Here are different types of pandas joins and how to use them in Python. Inner, Outer, Right, and Left joins are explained with examples from Amazon and Meta.

Pandas is an important Python tool to do data analysis and manipulation. Its typical use is working with data frames while analyzing and manipulating data. While working on different data frames, you can combine them using three different functions in four different ways.

How many Methods for Joining Data Frames are there in Pandas?

There are three types of join functions in Python.

1. **Merge**: It merges two data frames on columns or indices.
2. **Join**: The `join()` function will join two data frames together on a key column or index.

3. **Concat**: The `concat()` function bonds two data frames across the rows or columns.

It sounds rather similar, so what is the difference between these three approaches?

`Merge()` allows you to perform more flexible table joins because it provides you more combinations, yet `concat()` is less structured. `Join()` combines data frames on the index but not on columns, yet `merge()` gives you a chance to specify the column you want to join on.

In our examples, we will use `merge()` to show you how different types of joins in python work.

What's the Point of Joins in Pandas?

In simple terms, pandas joins in python are used to combine two data frames. When doing that, you have to specify the type of join. The defined pandas join type specifies how the data frames will be joined.

Now, let's look at the types of python pandas joins with the merge function.

Types of Pandas Joins in Python?

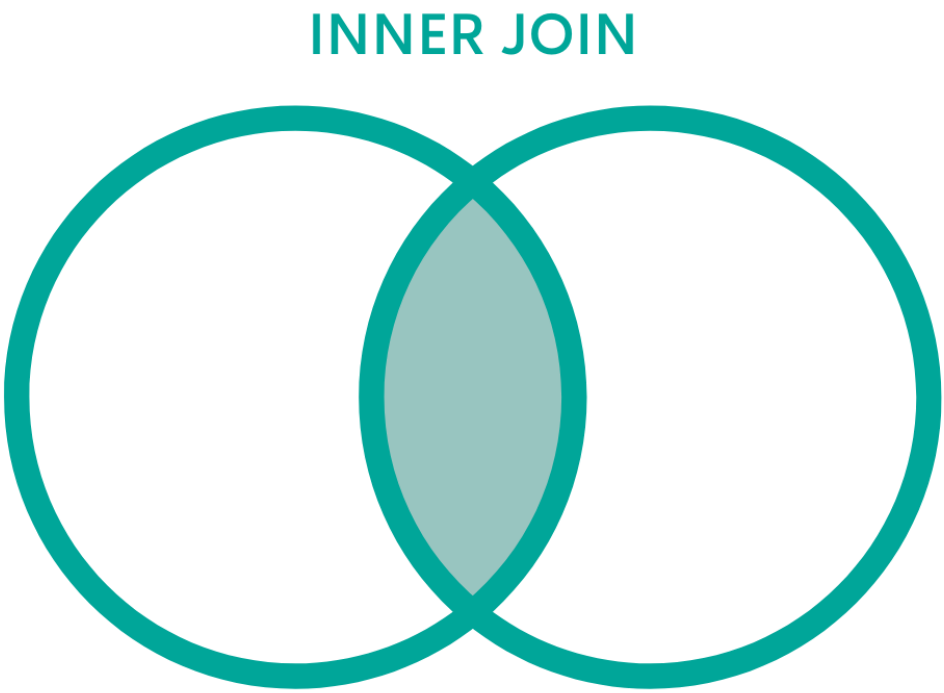
There are four main types of pandas joins in python, which we will explain in this article.

- Inner
- Outer
- Left
- Right

Here is the official page of the Python Pandas “merge” function, which we will use to join two data frames.

Now, let’s get started with Inner Join.

Inner Join in Python



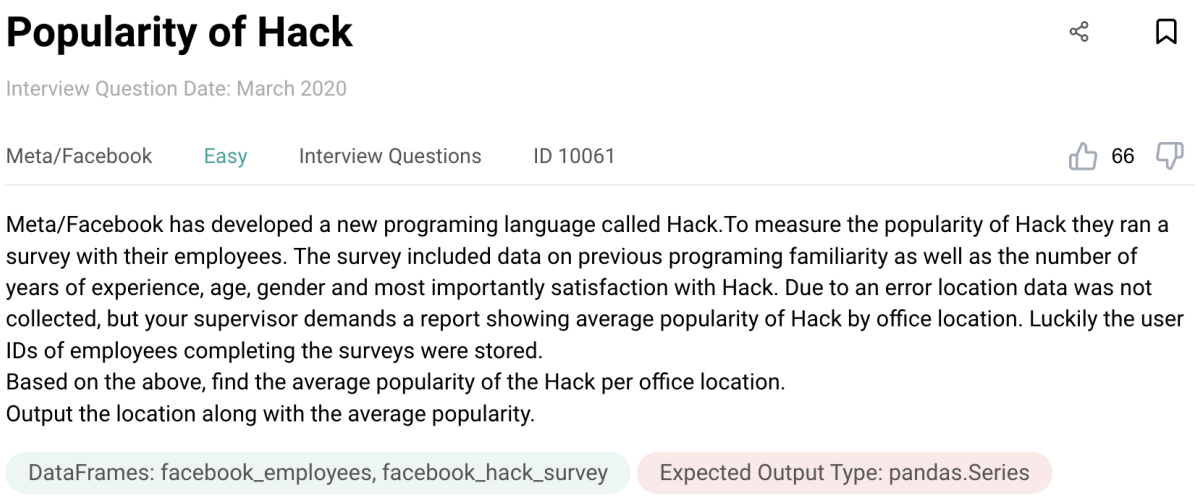
What is Inner Join in Python?

Inner join in pandas is used to merge two data frames at the intersection. It returns the mutual rows of both.

Inner Join Example

Now, let's look at the example from the platform. Our Inner Join question from the Meta(Facebook).

Meta developed a new programming language called Hack. To measure its popularity, they ran a survey with their employees. Due to an error location, data was not collected, but your supervisor demands a report showing the average popularity of Hach by office location.
Now the aim is to find the average popularity of the Hack per office location.
Output has to contain the location along with the average popularity.



Link to the question: https://platform.stratascratch.com/coding/10061-popularity-of-hack

Data
We have two data frames. Our first data frame is **facebook_employees.** The table has the following columns.

The data preview is shown below.

-

id	location	age	gender	is_senior
0	USA	24	M	FALSE
1	USA	31	F	TRUE
2	USA	29	F	FALSE
3	USA	33	M	FALSE
4	USA	36	F	TRUE

The second data frame is **facebook_hack_survey,** and it has the following columns.

facebook_hack_survey

Preview

employee_id:	int64
age:	int64
gender:	object
popularity:	int64

Also, the data preview is shown below.

employee_id	age	gender	popularity
0	24	M	6
1	31	F	4
2	29	F	0
3	33	M	7
4	36	F	6

Solution Approach

1. Let's load the libraries.
2. Now, we have to merge two data frames to find the popularity of the location.
3. Since the question wants us to show popularity and location, we will group by two columns, and then we will use the `mean()` function to find the average and `reset_index()` to remove indexes that the `groupby()` function creates.

Coding

1. Let's import the NumPy and Pandas libraries first to manipulate the data and use the statistical methods with it.

```
import pandas as pd
import numpy as np
```

If you want to know how to import pandas as pd in python and its importance for doing data science, check out our article "*How to Import Pandas as pd in Python*".

2. Now, question asks us to return to a location with popularity.

We have the location in the first data frame and the popularity in our second data frame. So to draw popularity and location together, let's merge two data frames using the inner join on `id`. The `age` and `gender` columns are in common, yet the `id` column has a different name in both data frames. That's why we matched the `left_on` argument with `id` and the `right_on` argument with `employee_id`.

We want to find the popularity of the Hack per office location. So the location and the popularity should match, that's why we need the intersection, so we will use inner join.

Selecting the right python join type is crucial to get the correct answer. In this case, the left and inner join will return the same result. They will both return 14 rows, which are the commons of both tables.

Yet, the right join will return the whole right data frame, which contains 17 rows, and for the rest, there will be NA assigned on the left data frame.

Below is the info table of three data frames to see the information of the rows of the first, the second, and the merged data frames.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   id          14 non-null     int64
 1   location    14 non-null     object
 2   age         14 non-null     int64
 3   gender      14 non-null     object
 4   is_senior   14 non-null     bool
dtypes: bool(1), int64(2), object(2)
memory usage: 590.0+ bytes
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17 entries, 0 to 16
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   employee_id   17 non-null     int64
 1   age           17 non-null     int64
 2   gender        17 non-null     object
 3   popularity    17 non-null     int64
dtypes: int64(3), object(1)
memory usage: 672.0+ bytes
<class 'pandas.core.frame.DataFrame'>
Int64Index: 14 entries, 0 to 13
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   id            14 non-null     int64
 1   location      14 non-null     object
 2   age_x         14 non-null     int64
 3   gender_x      14 non-null     object
 4   is_senior     14 non-null     bool
 5   employee_id   14 non-null     int64
 6   age_y         14 non-null     int64
 7   gender_y      14 non-null     object
 8   popularity    14 non-null     int64
dtypes: bool(1), int64(5), object(3)
memory usage: 1022.0+ bytes
```

OK, let's get back to writing the answer using the inner join.

```
import pandas as pd
import numpy as np
merged = pd.merge(facebook_employees,facebook_hack_survey, left_on = 'id',
right_on = 'employee_id', how = 'inner')
```

3. The question wants us to return the average popularity based on the location, so let's use the `groupby()` function with `mean()` and reset indexes that `groupby()` creates using the `reset_index()` method.

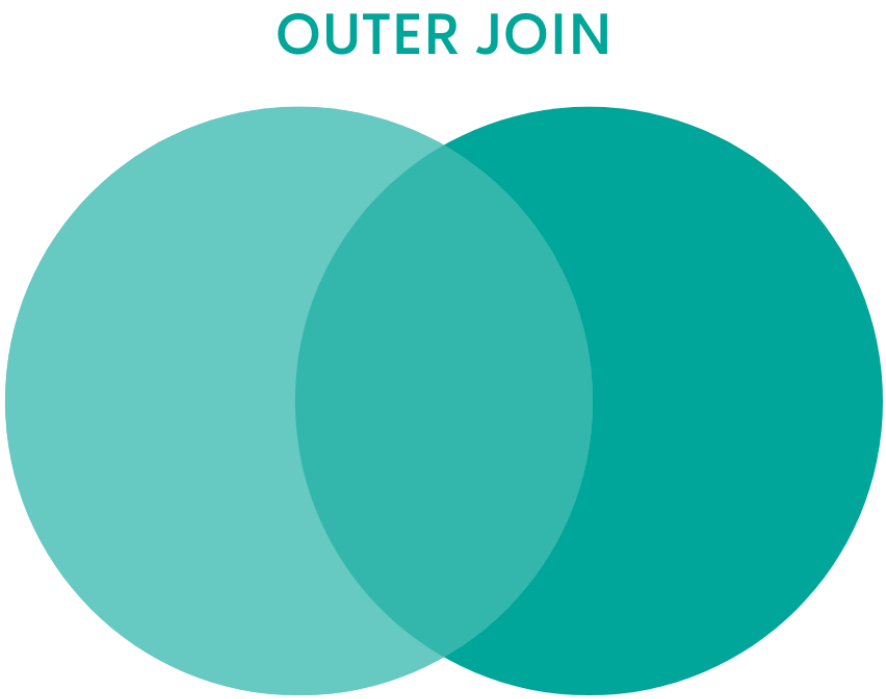
```
import pandas as pd
import numpy as np

merged = pd.merge(facebook_employees,facebook_hack_survey, left_on = 'id',
right_on = 'employee_id', how = 'inner')
result = merged.groupby(['location'])['popularity'].mean().reset_index()
```

Output
Here is the output, the average popularity based on the locations.

location	popularity
India	7.5
Switzerland	1
UK	4.333
USA	4.6

By the way, if you want to learn more about Pandas, here are Pandas Interview Questions for Data Science.

Outer Join in Python



What is Outer Join in Python?

It might also be called Full Outer Join and returns all rows from both DataFrames. It will match all rows that exist in both data frames. The rows found only in one of the two DataFrames will get the NA value.

Outer Join Example

Now, let's look at an example of Outer join from the platform.

An app has product features to help guide users through a marketing funnel. Each funnel has steps as a guide to complete the funnel. Meta asks us to find the average percentage of completion for each feature.

User Feature Completion

Meta/Facebook Medium General Practice ID 9792 13

An app has product features that help guide users through a marketing funnel. Each feature has "steps" (i.e., actions users can take) as a guide to complete the funnel. What is the average percentage of completion for each feature?

DataFrames: facebook_product_features, facebook_product_features_realizations

Link to the question: https://platform.stratascratch.com/coding/9792-user-feature-completion

Data
We have two data frames. Our first data frame is **facebook_product_features**. The data frame has the following columns.

facebook_product_features Preview

feature_id:	int64
n_steps:	int64

Also, the data preview is shown below.

feature_id	n_steps
0	5
1	7
2	3

The second data frame is **facebook_product_features_realizations** with the following columns.

facebook_product_features_realizations

Preview

feature_id:	int64
user_id:	int64
step_reached:	int64
timestamp:	datetime64[ns]

The data preview is shown below.

feature_id	user_id	step_reached	timestamp
0	0	1	2019-03-11 17:15:00
0	0	2	2019-03-11 17:22:00
0	0	3	2019-03-11 17:25:00
0	0	4	2019-03-11 17:27:00
0	1	1	2019-03-11 19:51:00

Solution Approach

1. Load the pandas library.
2. Group by the `feature_id` and `user id`, and calculate the max step reached.
3. Merge two data frames on `feature_id` using the outer join and fill NAs with zero.
4. Calculate the share of completion by dividing the step reached with `n_step` times 100 to find the percentage.
5. Group the data frame by `feature_id` and select the share of completion, calculate the mean, reset the index, and save the results to frame.

Coding

1. Let’s import the pandas library first to manipulate the data.

```
import pandas as pd
```

2. Now here is the time to find the maximum step by grouping by the `feature_id` and `user_id` first. Then select the step reached and use the `max()` function afterward. After that, we will reset the index that the `groupby()` function creates.

```
import pandas as pd

max_step = facebook_product_features_realizations.groupby(["feature_id",
"user_id"])[
   "step_reached"].max().reset_index()
```

3. Next, we have to calculate the share of completion. We will divide the step reached by n_steps and multiply by 100. So we have to select `n_steps` from the first data frame and `step_reached` from the second data frame. We will combine them on `feature_id` using the outer join because we need all values from both data frames to do the math. The non-matching values will be NA, so we will replace these values with zero after merging.

```
import pandas as pd

max_step = facebook_product_features_realizations.groupby(["feature_id",
"user_id"])[
   "step_reached"].max().reset_index()
df = pd.merge(facebook_product_features, max_step, how='outer',
on='feature_id').fillna(0)
```

4. At this stage, we will calculate the share of completion by dividing the step reached by the number of steps and multiplying by 100.

```
import pandas as pd

max_step = facebook_product_features_realizations.groupby(["feature_id",
"user_id"])[
   "step_reached"].max().reset_index()
df = pd.merge(facebook_product_features, max_step, how='outer',
on='feature_id').fillna(0)
df["share_of_completion"] = (df["step_reached"] / df["n_steps"])*100
```

5. Now, we will group by the data frame by `feature_id`, select the share of completion and calculate the mean. Then we will assign these results to the column `avg_share_of_completion` and reset the index.

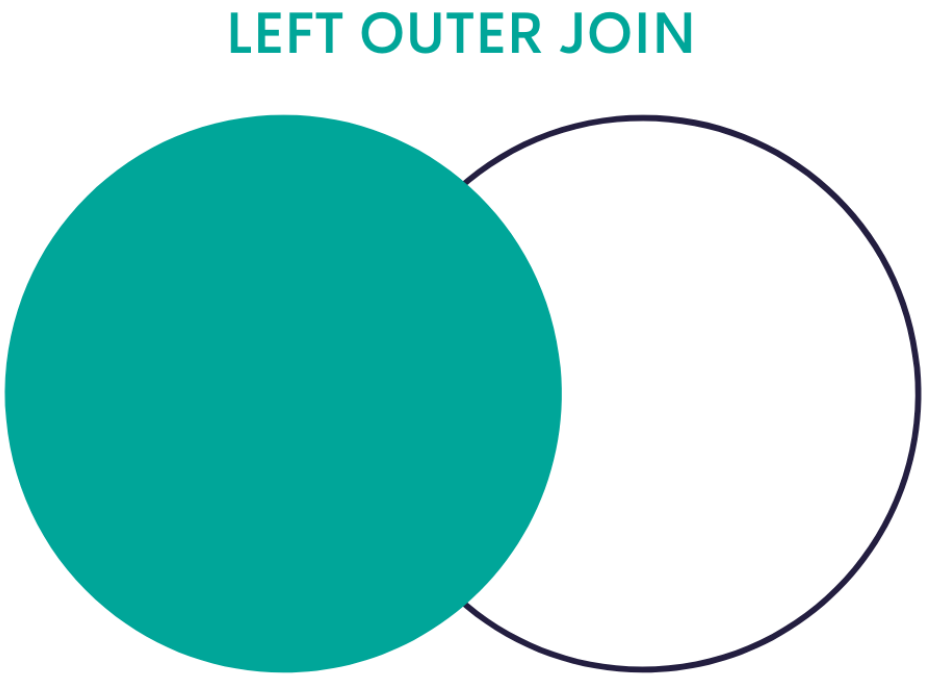
```
import pandas as pd

max_step = facebook_product_features_realizations.groupby(["feature_id",
"user_id"])[
   "step_reached"].max().reset_index()
df = pd.merge(facebook_product_features, max_step, how='outer',
on='feature_id').fillna(0)
df["share_of_completion"] = (df["step_reached"] / df["n_steps"])*100
result =
df.groupby("feature_id")["share_of_completion"].mean().to_frame("avg_share_
of_completion").reset_index()
```

Output
Here is the expected output.

feature_id	avg_share_of_completion
0	80
1	76.19
2	0

If you want to enhance your Python skills, here are Python Interview Questions and Answers.

Left Outer Join in Python



What is Left Outer Join in Python?

The left outer join returns all rows from the left data frame, which will be merged with the corresponding rows from the right data frame. It fills the unmatched rows with NA.

Left Outer Join Example

This time, our coding question is from Amazon.

Amazon asks us to sort records based on the customer's first name and the order details in ascending order.

Customer Details

Interview Question Date: April 2019

Amazon Easy Interview Questions ID 9891 68

Find the details of each customer regardless of whether the customer made an order. Output the customer's first name, last name, and the city along with the order details.
You may have duplicate rows in your results due to a customer ordering several of the same items. Sort records based on the customer's first name and the order details in ascending order.

DataFrames: customers, orders | Expected Output Type: pandas.DataFrame

Link to the question: https://platform.stratascratch.com/coding/9891-customer-details

Data
We have two data frames. The first data frame is **customers.** The data frame has the following columns.

customers Preview

id:	int64
first_name:	object
last_name:	object
city:	object
address:	object
phone_number:	object

The data preview is shown below.

id	first_name	last_name	city	address	phone_number
8	John	Joseph	San Francisco		928-386-8164
7	Jill	Michael	Austin		813-297-0692
4	William	Daniel	Denver		813-368-1200
5	Henry	Jackson	Miami		808-601-7513
13	Emma	Isaac	Miami		808-690-5201

Our second data frame is **orders**. The data frame has the following columns.

orders Preview

id:	int64
cust_id:	int64
order_date:	datetime64[ns]
order_details:	object
total_order_cost:	int64

Also, the data preview is shown below.

id	cust_id	order_date	order_details	total_order_cost
1	3	2019-03-04 00:00:00	Coat	100
2	3	2019-03-01 00:00:00	Shoes	80
3	3	2019-03-07 00:00:00	Skirt	30
4	7	2019-02-01 00:00:00	Coat	25
5	7	2019-03-10 00:00:00	Shoes	80

Solution Approach

1. Let's load the libraries.
2. Merge the data frames from the left on `id` and `cust_id`.
3. Sort values by the first names and order details and show `first_name`, `last_name`, `city`, and `order_details` in the output.

Coding

1. Now first, let's import pandas and NumPy libraries.

```
import pandas as pd
import numpy as np
```

2. Here, we will merge both data frames using the left join because the output should contain the sorted records based on the customer's first name and the order details. We need the list of all customers.

```
import pandas as pd
import numpy as np

merged = pd.merge(customers, orders, left_on = 'id', right_on = 'cust_id',
how = 'left')
```

3. It is time to sort values according to the first name and order details and select the first name, last name, city, and order details.

```
import pandas as pd
import numpy as np
```

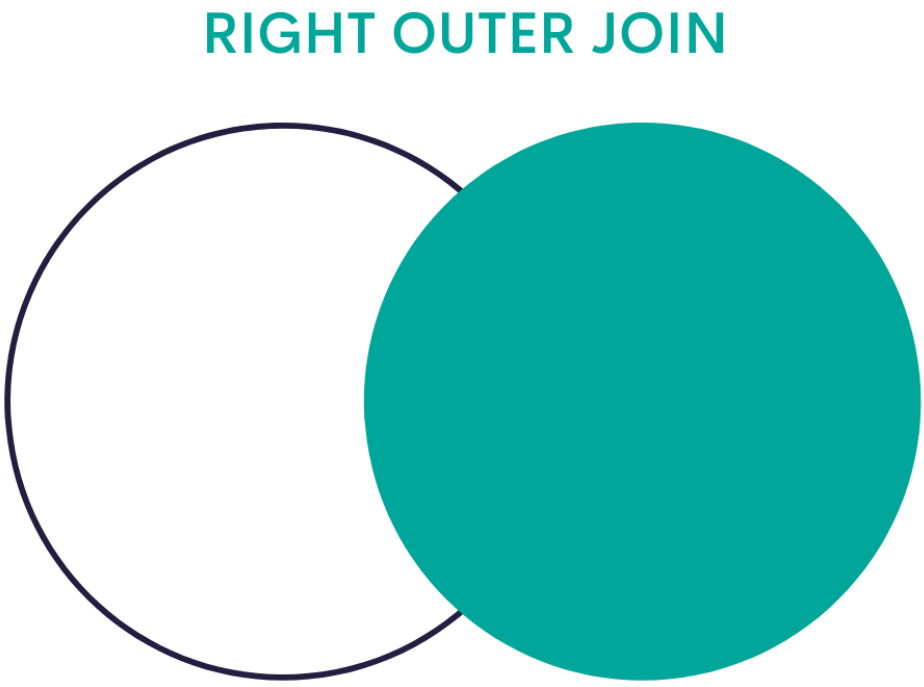
```
merged = pd.merge(customers, orders, left_on = 'id', right_on = 'cust_id',
how = 'left')
result =
merged[['first_name','last_name','city','order_details']].sort_values(['fir
st_name','order_details'])
```

Output

Here is the expected output.

first_name	last_name	city	order_details
Emma	Isaac	Miami	
Eva	Lucas	Arizona	Coat
Eva	Lucas	Arizona	Shirts
Eva	Lucas	Arizona	Slipper
Farida	Joseph	San Francisco	Coat
Farida	Joseph	San Francisco	Shirts
Farida	Joseph	San Francisco	Shoes
Farida	Joseph	San Francisco	Skirt
Frank	Jacob	Miami	
Henry	Jackson	Miami	Shoes
Jack	Aiden	Arizona	
Jill	Michael	Austin	Coat
Jill	Michael	Austin	Coat
Jill	Michael	Austin	Coat
Jill	Michael	Austin	Dresses
Jill	Michael	Austin	Shoes

Right Outer Join in Python



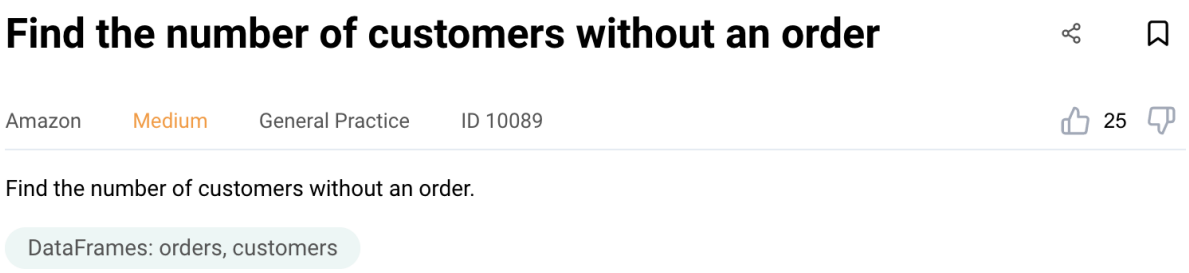
What is Right Outer Join in Python?

The right outer join returns all rows from the right data frame and the remaining data from the left. The data which does not correspond to the right data frame will have NAs assigned.

Right Outer Join Example

This question is from Amazon.

Amazon asks us to find the number of customers without an order.



Link to the question:
https://platform.stratascratch.com/coding/10089-find-the-number-of-customers-without-an-order

Data
We have two data frames.The **orders** data frame has the following columns.

orders Preview

id:	int64
cust_id:	int64
order_date:	datetime64[ns]
order_details:	object
total_order_cost:	int64

Here is the preview of the data.

id	cust_id	order_date	order_details	total_order_cost
1	3	2019-03-04 00:00:00	Coat	100
2	3	2019-03-01 00:00:00	Shoes	80
3	3	2019-03-07 00:00:00	Skirt	30
4	7	2019-02-01 00:00:00	Coat	25
5	7	2019-03-10 00:00:00	Shoes	80

The second data frame is **customers.** The data frame has the following columns.

customers

Preview

id:	int64
first_name:	object
last_name:	object
city:	object
address:	object
phone_number:	object

Also, here is the preview.

id	first_name	last_name	city	address	phone_number
8	John	Joseph	San Francisco		928-386-8164
7	Jill	Michael	Austin		813-297-0692
4	William	Daniel	Denver		813-368-1200
5	Henry	Jackson	Miami		808-601-7513
13	Emma	Isaac	Miami		808-690-5201

Solution Approach

1. First, let's load the libraries.
2. Merge two data frames from the right.
3. Find the customers without an order by using the `is_null()` method.
4. Find the number of customers without an order by using the `len()` method.

Coding

1. Let's import the NumPy and pandas libraries first to manipulate the data and use the statistical methods with it.

```
import pandas as pd
import numpy as np
```

2. Now, we will merge these two data frames from the right to find the number of customers without an order. After merging two data frames from the right, the customer's order data column will be null if there's no order. And we can find the customers who haven't had any orders in the next step.

```
import pandas as pd
import numpy as np

merged =
pd.merge(orders,customers,left_on='cust_id',right_on='id',how='right')
```

3. Here we will use the `isnull()` function to find the customer ids that don't have any orders.

```
import pandas as pd
import numpy as np

merged =
pd.merge(orders,customers,left_on='cust_id',right_on='id',how='right')
null_cust = merged[merged['cust_id'].isnull()]
```

4. By using the `len()` function, we find the number of customers that have not had any orders.

```
import pandas as pd
import numpy as np

merged =
pd.merge(orders,customers,left_on='cust_id',right_on='id',how='right')
null_cust = merged[merged['cust_id'].isnull()]
result = len(null_cust)
```

Output

Here is the expected output.

If you want to discover the join in SQL too, here are Different Types of SQL JOINs.

Conclusion

In this article, you learned about four pandas joins in python through the interview questions by the companies like Meta and Amazon. These questions showed you how to use the joins in python and, more specifically, where to use them while doing data manipulation step by step.

Practicing similar interview questions will keep you ready for interviews. You should turn it into a habit. So join the StrataScratch community and sign up today to help us find your dream job.