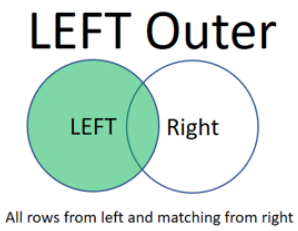
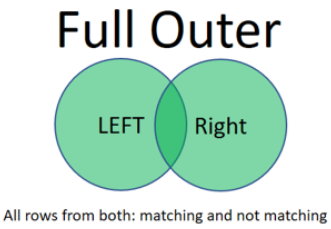
Joining, Filtering & Merging

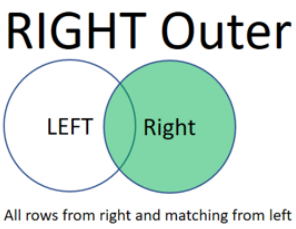
June 9, 2024

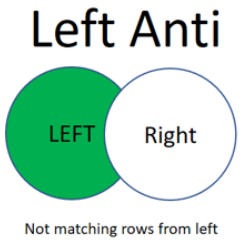
[1]:
```
import numpy as np
import pandas as pd
```

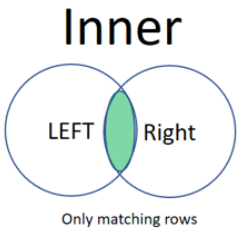
- Merging two table is called joining ## Types of joins

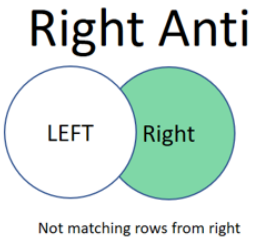






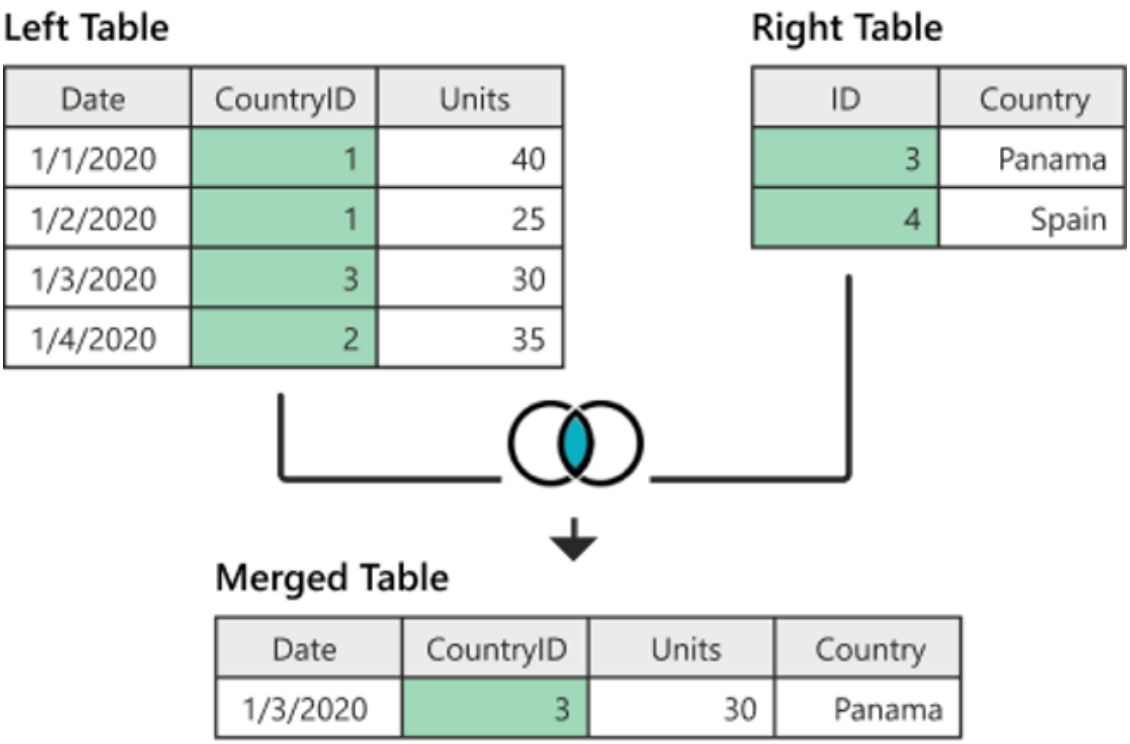






0.1 Inner Join

-Inner join will bring the same rows based on particular column



[2]:
```
inner_left_table = {'Date': ['1/1/2020', '1/2/2020', '1/3/2020', '1/4/2020'],
                    'CountryID': [1,1,3,2],
                    'Units': [40, 25, 30, 35]}
```

[3]:
```
inner_left_table_df = pd.DataFrame(inner_left_table)
inner_left_table_df
```

[3]:
```
        Date  CountryID  Units
0  1/1/2020          1     40
1  1/2/2020          1     25
2  1/3/2020          3     30
3  1/4/2020          2     35
```

[4]:
```
inner_right_table = {'ID': [3,4],
                     'Country': ['Panama', 'Spain']}
```

[5]:
```
inner_right_table_df = pd.DataFrame(inner_right_table)
inner_right_table_df
```

[5]:
```
   ID Country
0   3  Panama
1   4   Spain
```

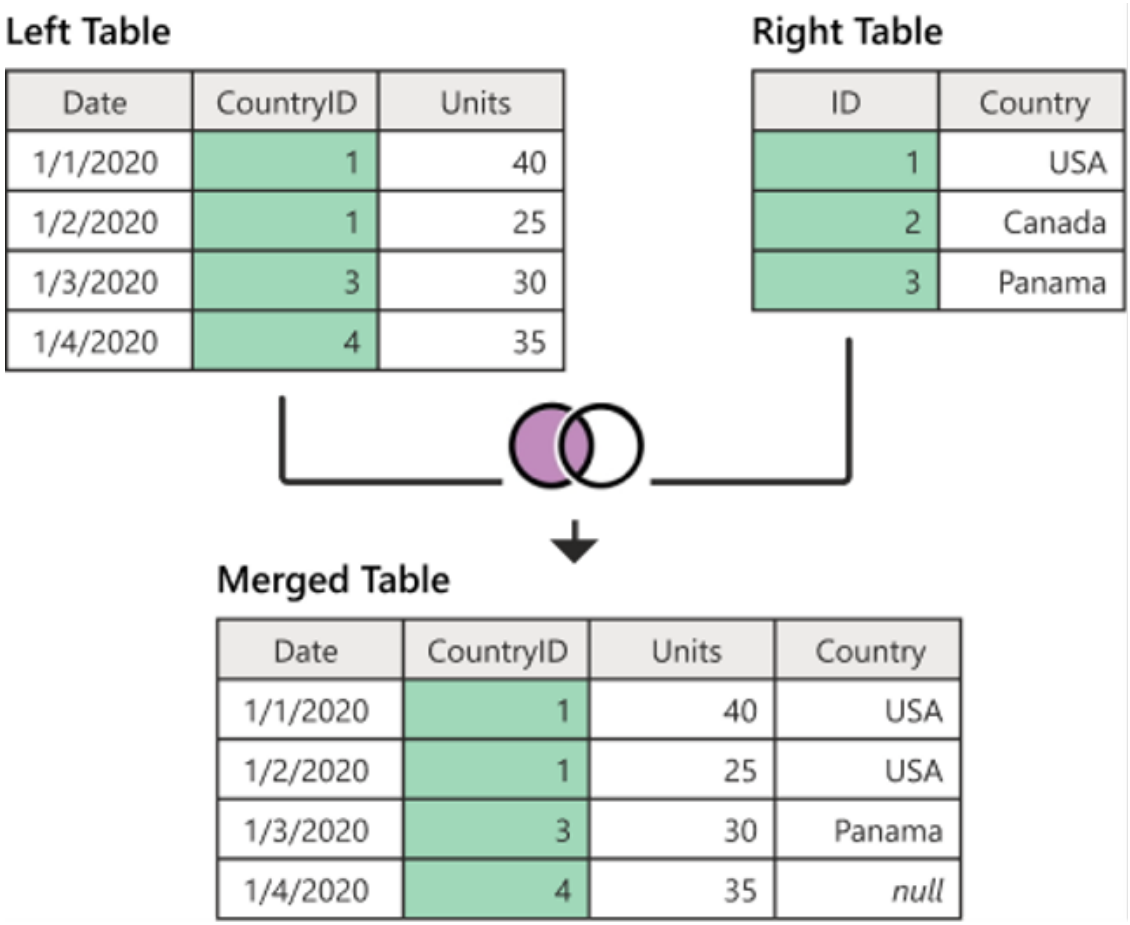
[6]:
```
inner_left_table_df.merge(inner_right_table_df, left_on='CountryID',␣
↪right_on='ID').drop('ID', axis=1) # It will perform inner merge by default
```

[6]:

	Date	CountryID	Units	Country
0	1/3/2020	3	30	Panama

0.2 Left / Left Outer Join

- It will bring all rows from left & matching rows from right.



[7]:
```
left_table_l = {'Date': ['1/1/2020', '1/2/2020', '1/3/2020', '1/4/2020'],
                'CountryID': [1,1,3,4],
                'Units': [40, 25, 30, 35]}
```

[8]:
```
left_table_l_df = pd.DataFrame(left_table_l) # left_table_l_df => left side table for left join
left_table_l_df
```

[8]:

	Date	CountryID	Units
0	1/1/2020	1	40
1	1/2/2020	1	25
2	1/3/2020	3	30
3	1/4/2020	4	35

[9]:
```
right_table_l = {'ID': [1,2,3],
                 'Country': ['USA', 'Canada', 'Panama']}
```

[10]:
```
right_table_l_df = pd.DataFrame(right_table_l) # right_table_l_df => right side table for left join
right_table_l_df
```

[10]:

	ID	Country
0	1	USA
1	2	Canada
2	3	Panama

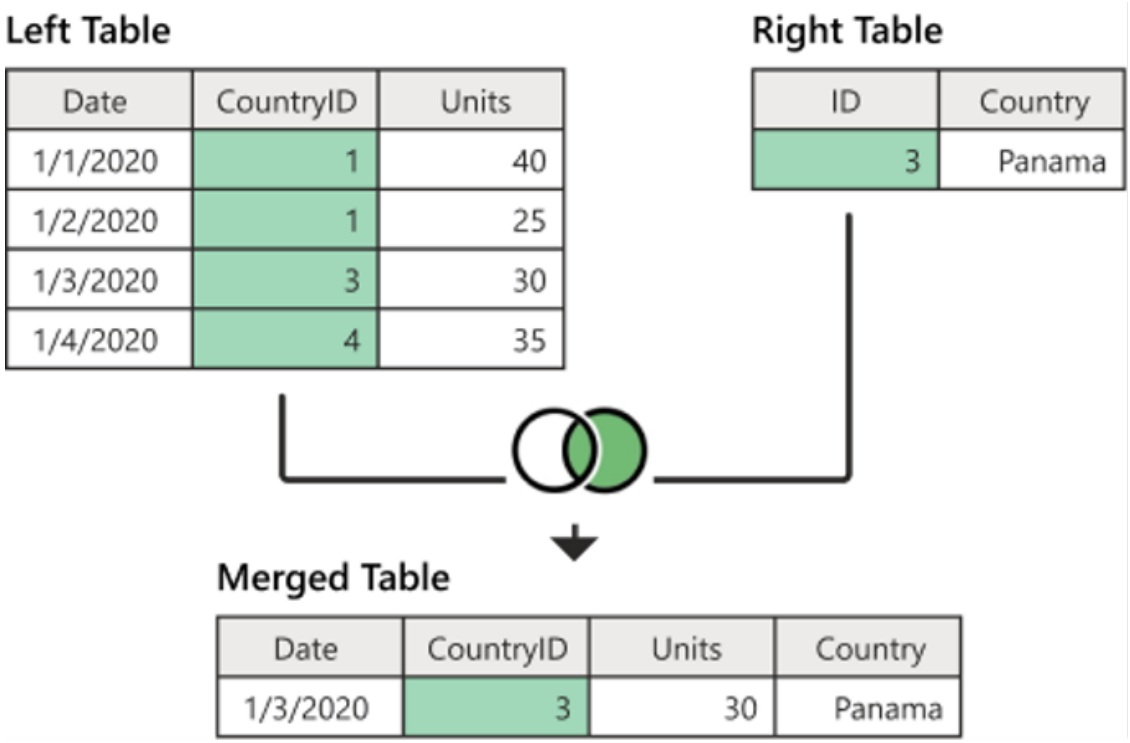
[11]:
```
left_table_l_df.merge(right_table_l_df, how='left', left_on='CountryID',right_on='ID').drop('ID', axis=1)
```

[11]:

	Date	CountryID	Units	Country
0	1/1/2020	1	40	USA
1	1/2/2020	1	25	USA
2	1/3/2020	3	30	Panama
3	1/4/2020	4	35	NaN

0.3 Right / Right Outer Join

- It will bring all rows from right & matching rows from left.



[12]:
```
left_table_r = {'Date': ['1/1/2020', '1/2/2020', '1/3/2020', '1/4/2020'],
                'CountryID': [1,1,3,4],
                'Units': [40, 25, 30, 35]}
```

[13]:
```
left_table_r_df = pd.DataFrame(left_table_r) # left_table_r_df => left table␣
 ↪for right join
left_table_r_df
```

[13]:

	Date	CountryID	Units
0	1/1/2020	1	40
1	1/2/2020	1	25
2	1/3/2020	3	30
3	1/4/2020	4	35

[14]:
```
right_table_r = {'ID': [3],
                'Country': ['Panama']}
```

[15]:
```
right_table_r_df = pd.DataFrame(right_table_r) # right_table_r_df => right␣
 ↪table for right join
right_table_r_df
```

[15]:

	ID	Country
0	3	Panama

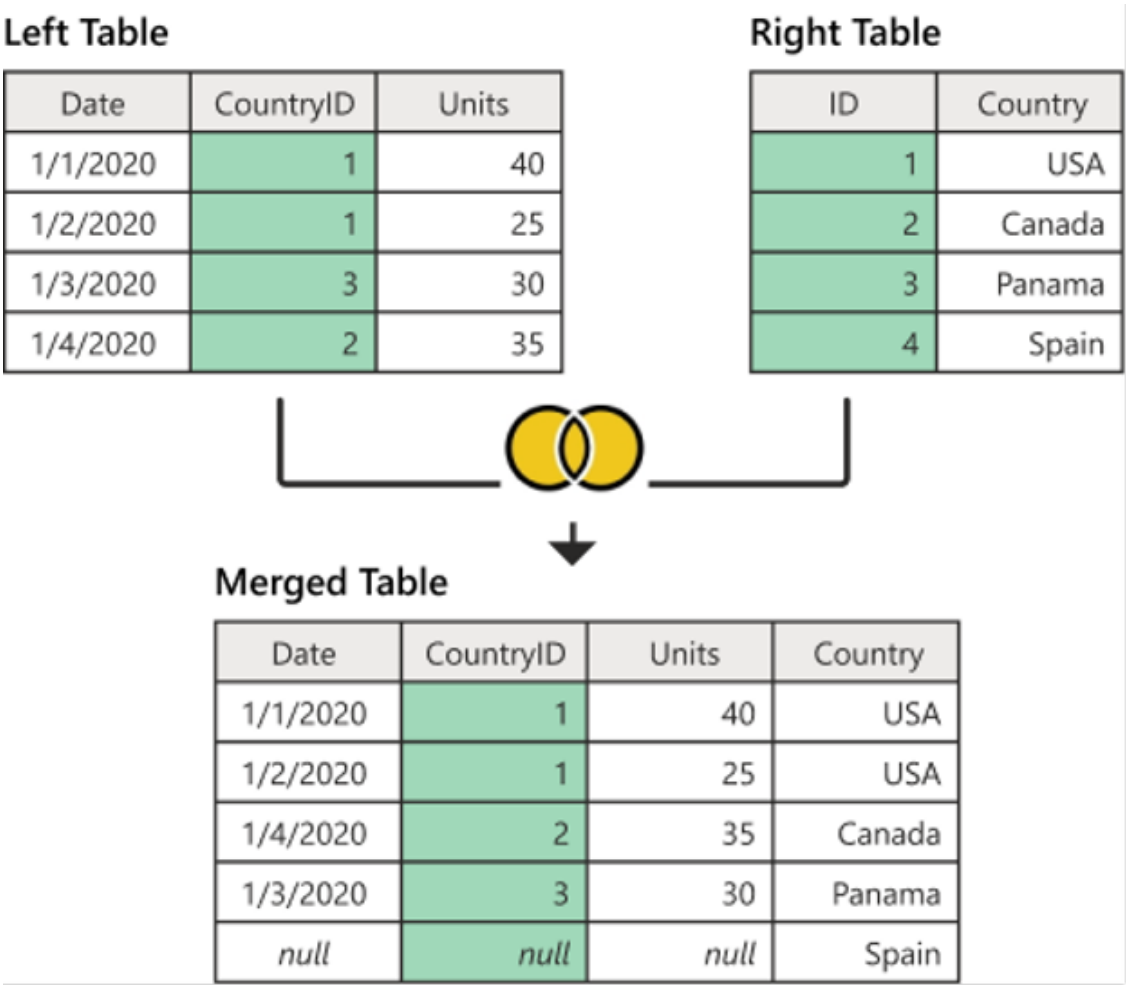
[16]:
```
left_table_r_df.merge(right_table_r_df, how='right', left_on='CountryID',␣
 ↪right_on='ID').drop('ID', axis=1)
```

[16]:

	Date	CountryID	Units	Country
0	1/3/2020	3	30	Panama

0.4 Full/Outer Join

- All rows from both tables



[17]:
```
left_table_f = {'Date': ['1/1/2020', '1/2/2020', '1/3/2020', '1/4/2020'],
                'CountryID': [1,1,3,2],
                'Units': [40, 25, 30, 35]}
```

[18]:
```
left_table_f_df = pd.DataFrame(left_table_f) # left_table_f_df => left table␣
↪for full join
left_table_f_df
```

[18]:

	Date	CountryID	Units
0	1/1/2020	1	40
1	1/2/2020	1	25
2	1/3/2020	3	30
3	1/4/2020	2	35

[19]:
```
right_table_f = {
    'ID': [1, 2, 3, 4],
    'Country': ['USA', 'Canada', 'Panama', 'Spain']
}
```

[20]:
```
right_table_f_df = pd.DataFrame(right_table_f) # right_table_f_df => right␣
↪table for full join
right_table_f_df
```

[20]:

	ID	Country
0	1	USA
1	2	Canada
2	3	Panama
3	4	Spain

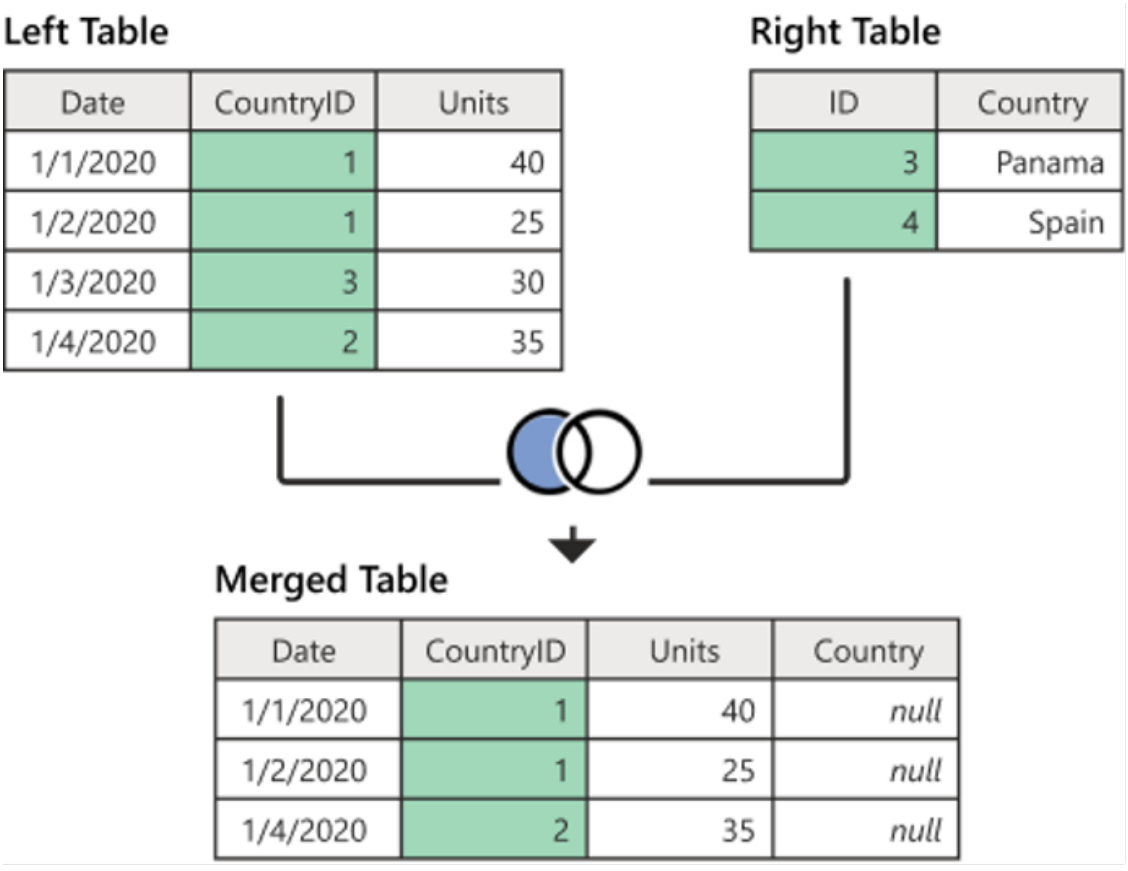
[21]:
```
left_table_f_df.merge(right_table_f_df, how='outer', left_on='CountryID',␣
↪right_on='ID').drop('ID', axis=1)
```

[21]:

	Date	CountryID	Units	Country
0	1/1/2020	1.0	40.0	USA
1	1/2/2020	1.0	25.0	USA
2	1/3/2020	3.0	30.0	Panama
3	1/4/2020	2.0	35.0	Canada
4	NaN	NaN	NaN	Spain

0.5 Left Anti Join

- Only Rows from left table that are not matching



[22]:
```
left_table_l_anti = {'Date': ['1/1/2020', '1/2/2020', '1/3/2020', '1/4/2020'],
                     'CountryID': [1,1,3,2],
                     'Units': [40, 25, 30, 35]}
```

[23]:
```
left_table_l_anti_df = pd.DataFrame(left_table_l_anti) #left_table_l_anti_df =>␣
 ↪left table for left anti join
left_table_l_anti_df
```

[23]:

	Date	CountryID	Units
0	1/1/2020	1	40
1	1/2/2020	1	25
2	1/3/2020	3	30
3	1/4/2020	2	35

[24]:
```
right_table_l_anti = {'ID': [3,4],
                      'Country': ['Panama', 'Spain']}
```

[25]:
```
right_table_l_anti_df = pd.DataFrame(right_table_l_anti) #␣
 ↪right_table_l_anti_df => right table for left anti join
right_table_l_anti_df
```

[25]:

	ID	Country
0	3	Panama
1	4	Spain

[26]:
```
left_anti_df = left_table_l_anti_df.merge(right_table_l_anti_df, how='left',␣
 ↪left_on='CountryID', right_on='ID', indicator=True).drop('ID', axis=1)
left_anti_df
```

[26]:

	Date	CountryID	Units	Country	_merge
0	1/1/2020	1	40	NaN	left_only
1	1/2/2020	1	25	NaN	left_only
2	1/3/2020	3	30	Panama	both
3	1/4/2020	2	35	NaN	left_only

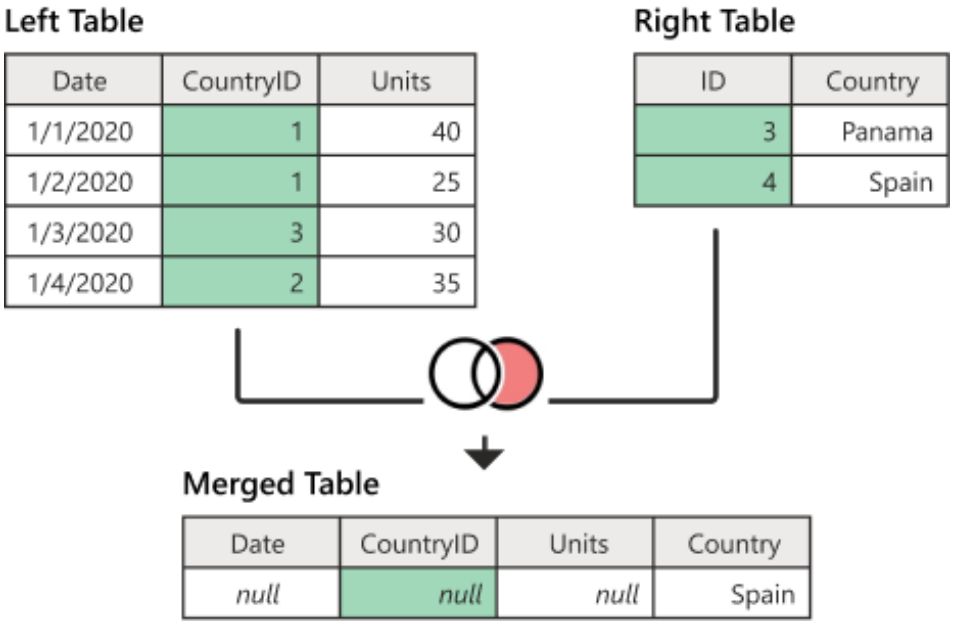
[27]:
```
left_anti_df[left_anti_df['_merge'] == 'left_only'].drop('_merge', axis=1)
```

[27]:

	Date	CountryID	Units	Country
0	1/1/2020	1	40	NaN
1	1/2/2020	1	25	NaN
3	1/4/2020	2	35	NaN

0.6 Right Anti Join

- All rows from right table that are not matching with left table



Date	CountryID	Units
1/1/2020	1	40
1/2/2020	1	25
1/3/2020	3	30
1/4/2020	2	35

ID	Country
3	Panama
4	Spain

Date	CountryID	Units	Country
null	*null*	*null*	Spain

[28]:
```
left_table_r_anti = {'Date': ['1/1/2020', '1/2/2020', '1/3/2020', '1/4/2020'],
                     'CountryID': [1,1,3,2],
                     'Units': [40, 25, 30, 35]}
```

[29]:
```
left_table_r_anti_df = pd.DataFrame(left_table_r_anti) # left_table_r_anti_df␣
 ↪=> Left table for right anti join
left_table_r_anti_df
```

[29]:

	Date	CountryID	Units
0	1/1/2020	1	40
1	1/2/2020	1	25
2	1/3/2020	3	30
3	1/4/2020	2	35

[30]:
```
right_table_r_anti = {'ID': [3,4],
                      'Country': ['Panama', 'Spain']}
```

[31]:
```
right_table_r_anti_df = pd.DataFrame(right_table_r_anti) #␣
 ↪right_table_r_anti_df => right table for right anti join
right_table_r_anti_df
```

[31]:

	ID	Country
0	3	Panama
1	4	Spain

- STEP:1 For right anti simply merge left table with right join with params indicator=True-STEP:2 Create a filter for left_ony row
- STEP:3 Apply that filter on whole datasetet

[32]:
```
right_anti_df = left_table_r_anti_df.merge(right_table_r_anti_df, how='right',␣
↪left_on='CountryID', right_on='ID', indicator=True).drop('ID', axis=1)
right_anti_df
```

[32]:

	Date	CountryID	Units	Country	_merge
0	1/3/2020	3.0	30.0	Panama	both
1	NaN	NaN	NaN	Spain	right_only

[33]:
```
right_anti_df[right_anti_df['_merge'] == 'right_only'].drop('_merge', axis=1)
```

[33]:

	Date	CountryID	Units	Country
1	NaN	NaN	NaN	Spain

0.7 Left Semi Join

- Return the intersection, similar to an inner join
- Return only column from left table and not the right
- No duplicated

[34]:
```
left_table_l_semi = {'Date': ['1/1/2020', '1/2/2020', '1/3/2020', '1/4/2020'],
                     'CountryID': [1,1,3,2],
                     'Units': [40, 25, 30, 35]}
```

[35]:
```
left_table_l_semi_df = pd.DataFrame(left_table_l_semi) # left_table_l_semi_df␣
↪=> Left table for left semi join
left_table_l_semi_df
```

[35]:

	Date	CountryID	Units
0	1/1/2020	1	40
1	1/2/2020	1	25
2	1/3/2020	3	30
3	1/4/2020	2	35

[36]:
```
right_table_l_semi = {'ID': [1, 3, 4],
                      'Country': ['USA','Panama', 'Spain']}
```

[37]:
```
right_table_l_semi_df = pd.DataFrame(right_table_l_semi) #␣
↪right_table_l_semi_df => right table for left semi join
right_table_l_semi_df
```

[37]:

	ID	Country
0	1	USA
1	3	Panama
2	4	Spain

[38]:
```
left_semi_join =  left_table_l_semi_df.merge(right_table_l_semi_df,␣
↪left_on='CountryID', right_on='ID').drop('ID', axis=1)
left_semi_join
```

[38]:
```
        Date  CountryID  Units Country
0  1/1/2020          1     40     USA
1  1/2/2020          1     25     USA
2  1/3/2020          3     30  Panama
```

[39]:
```
left_semi_join[['Date', 'CountryID', 'Units']].
↪drop_duplicates(subset='CountryID') # To remove duplicates
```

[39]:
```
        Date  CountryID  Units
0  1/1/2020          1     40
2  1/3/2020          3     30
```

0.8 Right Semi Join

- Return the intersection, similar to an inner join
- Return only column from right table and not the left
- No duplicated

[40]:
```
left_table_r_semi = {'Date': ['1/1/2020', '1/2/2020', '1/3/2020', '1/4/2020'],
              'CountryID': [1,1,3,2],
              'Units': [40, 25, 30, 35]}
```

[41]:
```
left_table_r_semi_df = pd.DataFrame(left_table_r_semi) # left_table_r_semi_df␣
↪=> Left table for right semi join
left_table_r_semi_df
```

[41]:
```
        Date  CountryID  Units
0  1/1/2020          1     40
1  1/2/2020          1     25
2  1/3/2020          3     30
3  1/4/2020          2     35
```

[42]:
```
right_table_r_semi = {'ID': [1, 3, 4],
              'Country': ['USA','Panama', 'Spain']}
```

[43]:
```
right_table_r_semi_df = pd.DataFrame(right_table_r_semi) #␣
↪right_table_r_semi_df => right table for right semi join
right_table_r_semi_df
```

[43]:
```
   ID Country
0   1     USA
1   3  Panama
2   4   Spain
```

[44]:
```
right_semi_join = left_table_r_semi_df.merge(right_table_r_semi_df,␣
↪left_on='CountryID', right_on='ID').drop('CountryID', axis=1)
right_semi_join
```

[44]:

	Date	Units	ID	Country
0	1/1/2020	40	1	USA
1	1/2/2020	25	1	USA
2	1/3/2020	30	3	Panama

[45]:
```
right_semi_join[['ID', 'Country']].drop_duplicates('ID')
```

[45]:

	ID	Country
0	1	USA
2	3	Panama

0.9 Concatination

[56]:
```
left_anti_df
```

[56]:

	Date	CountryID	Units	Country	_merge
0	1/1/2020	1	40	NaN	left_only
1	1/2/2020	1	25	NaN	left_only
2	1/3/2020	3	30	Panama	both
3	1/4/2020	2	35	NaN	left_only

[57]:
```
right_anti_df
```

[57]:

	Date	CountryID	Units	Country	_merge
0	1/3/2020	3.0	30.0	Panama	both
1	NaN	NaN	NaN	Spain	right_only

[59]:
```
pd.concat([left_anti_df, right_anti_df], ignore_index=True)
```

[59]:

	Date	CountryID	Units	Country	_merge
0	1/1/2020	1.0	40.0	NaN	left_only
1	1/2/2020	1.0	25.0	NaN	left_only
2	1/3/2020	3.0	30.0	Panama	both
3	1/4/2020	2.0	35.0	NaN	left_only
4	1/3/2020	3.0	30.0	Panama	both
5	NaN	NaN	NaN	Spain	right_only