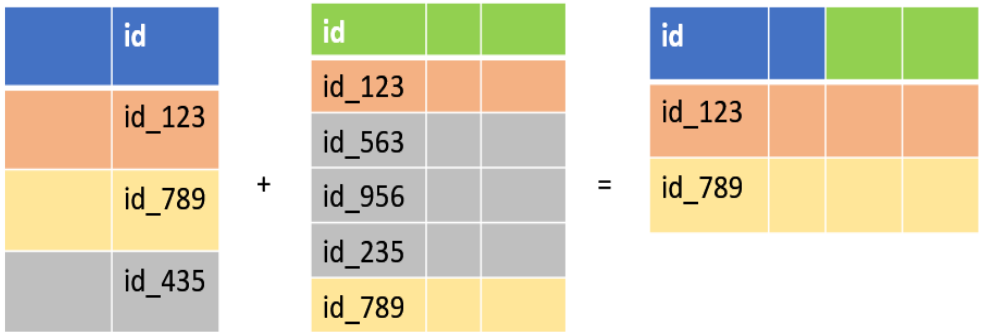
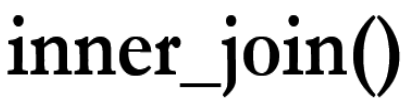
Inner Join

Method 3: inner_join()

Joining columns based on an *id* matching from two different dataframes. The resulting dataframe only has the rows in which there is an *id* in both dataframes, if not the rows are dropped. The id variable must be a character.





There is a dataframe named **basic_info_teen_f1_df** for teenagers that have the cd and downloaded versions of Game 1.

```
basic_info_teen_f1_df
## # A tibble: 3 x 4
##       id    group_age     age      gender
##    <chr>      <chr>    <dbl>      <chr>
## 1 id_333    teenage       17      Female
## 2 id_122    teenage       13      Female
## 3 id_530    teenage       14        Male
```

There is another dataframe named **basic_info_teen_f2_df** for teenages that have of Game 2.

```
basic_info_teen_f2_df
## # A tibble: 8 x 3
##       id    number_of_tournaments         level
##    <chr>                    <dbl>         <chr>
## 1 id_721                       23      competent
## 2 id_815                        9         expert
## 3 id_122                        2         expert
## 4 id_530                       12         novice
## 5 id_782                       10      competent
## 6 id_295                       24         novice
## 7 id_386                        1         novice
## 8 id_308                       17         novice
```

We can inner join **basic_info_teen_f1_df** and **basic_info_teen_f2_df** by using the following function and specifiying a 'by' variable,**id**, where order just places one dataframe in front of another:

```
inner_join(basic_info_teen_f1_df,basic_info_teen_f2_df, by = 'id')
## # A tibble: 2 x 6
##      id       group_age    age      gender    number_of_tournaments    level
##    <chr>        <chr>     <dbl>     <chr>             <dbl>           <chr>
## 1 id_122      teenage       13      Female             2              expert
## 2 id_530      teenage       14      Male               12             novice
```

This also could be done using the pipe function:

```
basic_info_teen_f1_df %>%
  inner_join(basic_info_teen_f2_df, by = 'id')
## # A tibble: 2 x 6
##      id       group_age    age      gender    number_of_tournaments    level
##    <chr>        <chr>     <dbl>     <chr>             <dbl>           <chr>
## 1 id_122      teenage       13      Female             2              expert
## 2 id_530      teenage       14      Male               12             novice
```

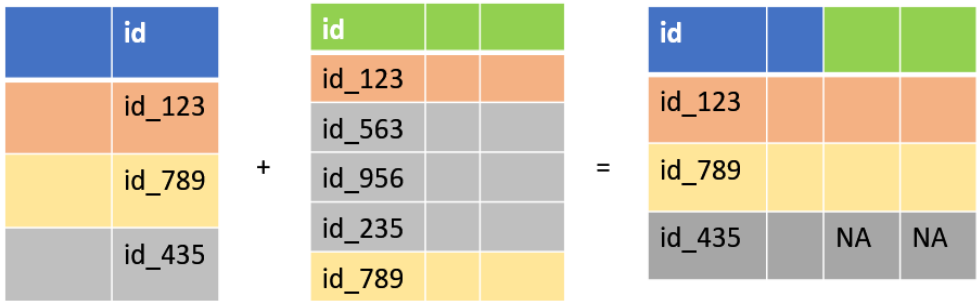
Question about Inner Join

There are two dataframes named, **basic_info_adult_f1_df** and **basic_info_adult_f2_df**. Combine the rows of these dataframes based on the common ids, provide an appropriate name for the dataframe, and specifiy the variable types in the combined dataframe.

Left Join

Method 4: left_join()

Joining columns based on an *id* matching from two different dataframes, however left_join keeps all rows from the left dataframe. The resulting dataframe only has the rows from the left dataframe and information from the right dataframe where the id matches. When the *id* from both dataframes does not match, NA is given. The id variable must be a character.



left_join()

There is a dataframe named **basic_info_teen_f1_df** for teenagers that have the cd and downloaded versions of Game 1.

```
basic_info_teen_f1_df
## # A tibble: 3 x 4
##       id    group_age     age      gender
##    <chr>      <chr>    <dbl>      <chr>
## 1 id_333    teenage       17      Female
## 2 id_122    teenage       13      Female
## 3 id_530    teenage       14        Male
```

There is another dataframe named **basic_info_teen_f2_df** for teenages that have of Game 2.

```
basic_info_teen_f2_df
## # A tibble: 8 x 3
##       id    number_of_tournaments         level
##    <chr>                    <dbl>         <chr>
## 1 id_721                       23      competent
## 2 id_815                        9         expert
## 3 id_122                        2         expert
## 4 id_530                       12         novice
## 5 id_782                       10      competent
## 6 id_295                       24         novice
## 7 id_386                        1         novice
## 8 id_308                       17         novice
```

We can left join **basic_info_teen_f1_df** and **basic_info_teen_f2_df** by using the following function and specifiying a 'by' variable,**id**, where order determines which is the left dataframe (**BE CAREFUL**):

```
left_join(basic_info_teen_f1_df,basic_info_teen_f2_df, by = 'id')
## # A tibble: 3 x 6
##      id      group_age    age    gender    number_of_tournaments    level
##   <chr>       <chr>     <dbl>   <chr>              <dbl>            <chr>
## 1 id_333    teenage      17     Female              NA              <NA>
## 2 id_122    teenage      13     Female              2               expert
## 3 id_530    teenage      14     Male                12              novice
```

This also could be done using the pipe function:

```
basic_info_teen_f1_df %>%
  left_join(basic_info_teen_f2_df, by = 'id')
## # A tibble: 3 x 6
##      id      group_age    age    gender    number_of_tournaments    level
##   <chr>       <chr>     <dbl>   <chr>              <dbl>            <chr>
## 1 id_333    teenage      17     Female              NA              <NA>
## 2 id_122    teenage      13     Female              2               expert
## 3 id_530    teenage      14     Male                12              novice
```

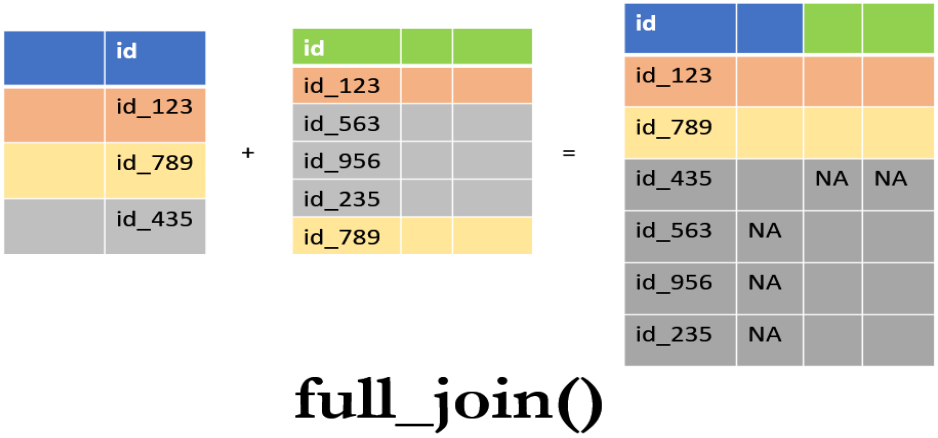
Question about Left Join

There are two dataframes named, **basic_info_adult_f1_df** and **basic_info_adult_f2_df**. Combine the rows of these dataframes based on the common ids where your resulting dataframe should have the rows from **basic_info_adult_f2_df**, provide an appropriate name for the dataframe, and specifiy the variable types in the combined dataframe.

Full Join

Method 5: full_join()

Joining columns based on an *id* matching from two different dataframes, however full_join keeps all rows from both dataframes. The resulting dataframe has the rows from the left and right dataframes however, creates 1 row for rows that match. When the *id* from both dataframes does not match, NA is given. The id variable must be a character.



There is a dataframe named **basic_info_teen_f1_df** for teenagers that have the cd and downloaded versions of Game 1.

```
basic_info_teen_f1_df
## # A tibble: 3 x 4
##       id    group_age     age      gender
##    <chr>      <chr>    <dbl>      <chr>
## 1 id_333    teenage       17      Female
## 2 id_122    teenage       13      Female
## 3 id_530    teenage       14        Male
```

There is another dataframe named **basic_info_teen_f2_df** for teenages that have of Game 2.

```
basic_info_teen_f2_df
basic_info_teen_f2_df
## # A tibble: 8 x 3
##       id    number_of_tournaments         level
##    <chr>                    <dbl>         <chr>
## 1 id_721                       23     competent
## 2 id_815                        9        expert
## 3 id_122                        2        expert
## 4 id_530                       12        novice
## 5 id_782                       10     competent
## 6 id_295                       24        novice
## 7 id_386                        1        novice
## 8 id_308                       17        novice
```

We can full join **basic_info_teen_f1_df** and **basic_info_teen_f2_df** by using the following function and specifiying a 'by' variable,**id**, where order determines which is the left dataframe (**BE CAREFUL**):

```
full_join(basic_info_teen_f1_df,basic_info_teen_f2_df, by = 'id')
## # A tibble: 9 x 6
##   id      group_age   age   gender  number_of_tournaments  level
##   <chr>   <chr>     <dbl>   <chr>           <dbl>         <chr>
## 1 id_333  teenage     17    Female           NA           <NA>
## 2 id_122  teenage     13    Female           2            expert
## 3 id_530  teenage     14    Male             12           novice
## 4 id_721  <NA>        NA    <NA>             23           competent
## 5 id_815  <NA>        NA    <NA>             9            expert
## 6 id_782  <NA>        NA    <NA>             10           competent
## 7 id_295  <NA>        NA    <NA>             24           novice
## 8 id_386  <NA>        NA    <NA>             1            novice
## 9 id_308  <NA>        NA    <NA>             17           novice
```

This also could be done using the pipe function:

```
basic_info_teen_f1_df %>%
  full_join(basic_info_teen_f2_df, by = 'id')
## # A tibble: 9 x 6
##   id      group_age   age   gender  number_of_tournaments  level
##   <chr>   <chr>     <dbl>   <chr>           <dbl>         <chr>
## 1 id_333  teenage     17    Female           NA           <NA>
## 2 id_122  teenage     13    Female           2            expert
## 3 id_530  teenage     14    Male             12           novice
## 4 id_721  <NA>        NA    <NA>             23           competent
## 5 id_815  <NA>        NA    <NA>             9            expert
## 6 id_782  <NA>        NA    <NA>             10           competent
## 7 id_295  <NA>        NA    <NA>             24           novice
## 8 id_386  <NA>        NA    <NA>             1            novice
## 9 id_308  <NA>        NA    <NA>             17           novice
```

Question about Full Join

There are two dataframes named, **basic_info_adult_f1_df** and **basic_info_adult_f2_df**. Combine the rows of these dataframes based on the common ids where your resulting dataframe should have the rows from both original dataframes, provide an appropriate name for the dataframe, and specifiy the variable types in the combined dataframe.