Follow Us!

@bitprojectorg

@bitprj

@BitProject

@BitProject

join.bitproject.org

Bit Project

Visualizing LinkedIn Profiles and Jobs

Lab 1

Table of Contents

- Recap
- About the Data
- Labs and Methodology

Recap

By this time, you should be able to...

- Load a dataset '.csv' as a data frame using **pd.read_csv**
- Observe **properties** of a loaded dataset using:
    - pd.head()
    - pd.describe()
    - pd.info()

Recap

By this time, you should be able to...

- **Modify** the dataset (remove NaN values)
- Understand the term **object** in DataFrames
- **Re-index** columns
- Visualize data using **matplotlib** and **pandas**: scatter plots, barplots, line plots, histograms

About the Dataset

LinkedIn

- Employment-oriented online service
- Professional networking and development
- ~39,500 LinkedIn profiles
- Ages 20 to 86 years
- Profiles across various countries and major companies

Bit Project

Labs and Methodology

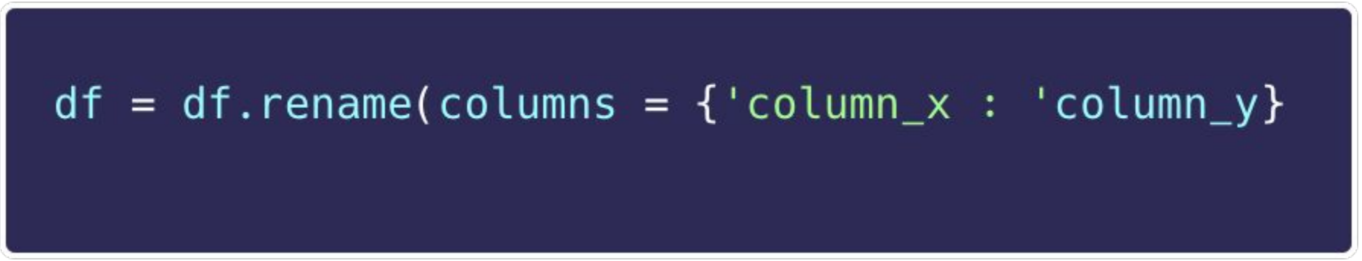
- **Purpose**:
  - **Apply** the Data Science pipeline in a project-based environment
  - **Use** the tools taught to you in the previous modules
- **Method:**
  - **Question and Answer** based approach

Labs and Methodology

Start by **asking questions** which you will **answer in code** and simple **explanations**.

Example

change the name of 'column_x' to 'column y'



```
df = df.rename(columns = {'column_x : 'column_y}
```

Labs and Methodology

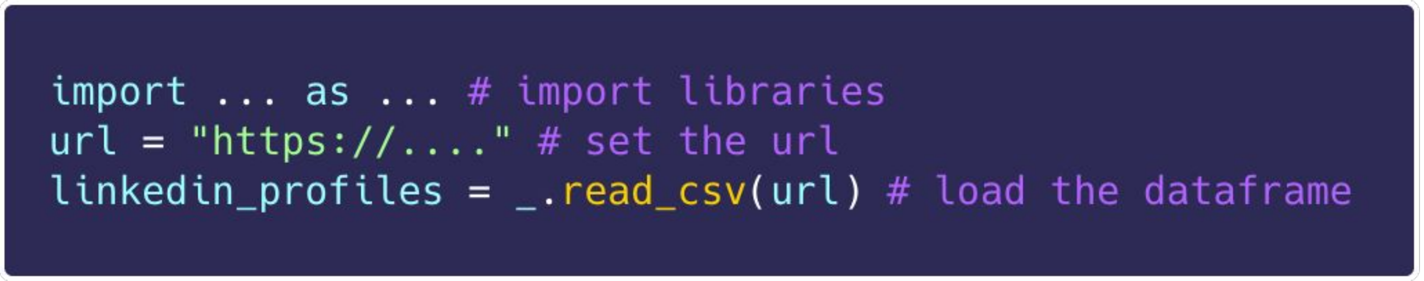
1. Set up
2. Choose columns
3. Split dataframe

Part 1
Setting Up Our Basic Data Analysis

Setting Up our Basic Data Analysis

Which **libraries** do we want to import?

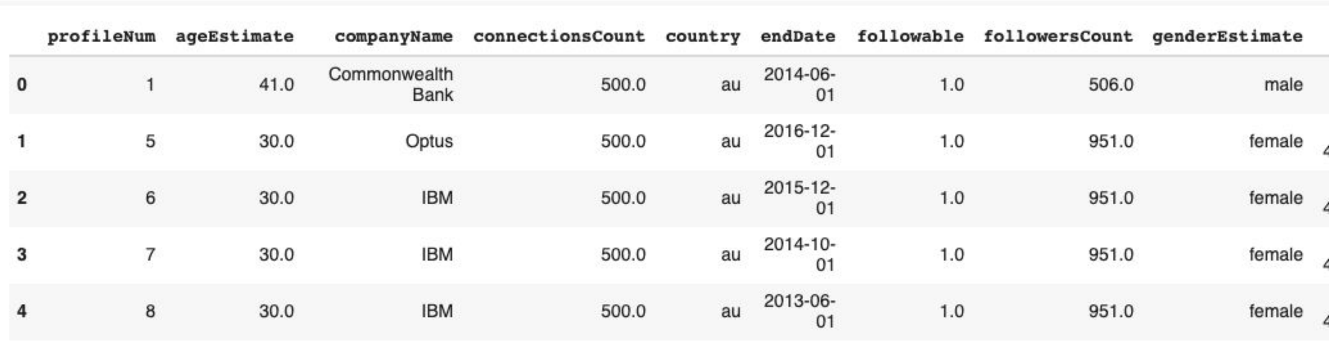
How do we **load** the dataframe?



```
import ... as ... # import libraries
url = "https://...." # set the url
linkedin_profiles = _.read_csv(url) # load the dataframe
```

head()

- Check what **columns** this file has
- Returns **first n rows** (by default, the first 5 rows)
- For dataframe df, you can specify the number of rows to display by calling df.head(number)

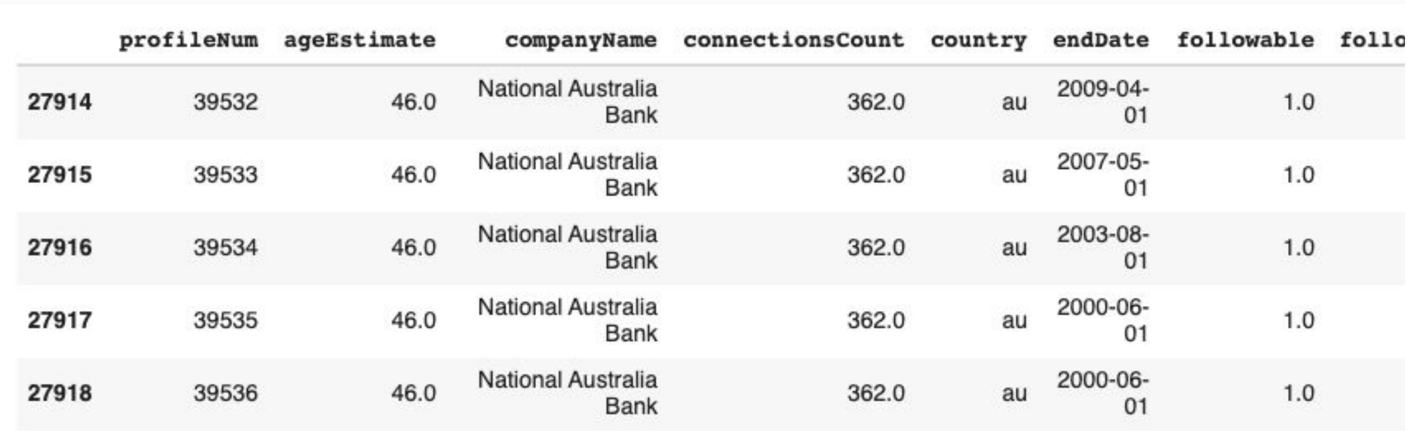


	profileNum	ageEstimate	companyName	connectionsCount	country	endDate	followable	followersCount	genderEstimate
0	1	41.0	Commonwealth Bank	500.0	au	2014-06-01	1.0	506.0	male
1	5	30.0	Optus	500.0	au	2016-12-01	1.0	951.0	female
2	6	30.0	IBM	500.0	au	2015-12-01	1.0	951.0	female
3	7	30.0	IBM	500.0	au	2014-10-01	1.0	951.0	female
4	8	30.0	IBM	500.0	au	2013-06-01	1.0	951.0	female

By default we have the first 5 rows

tail()

- Prints the **last n rows** of our dataset by default

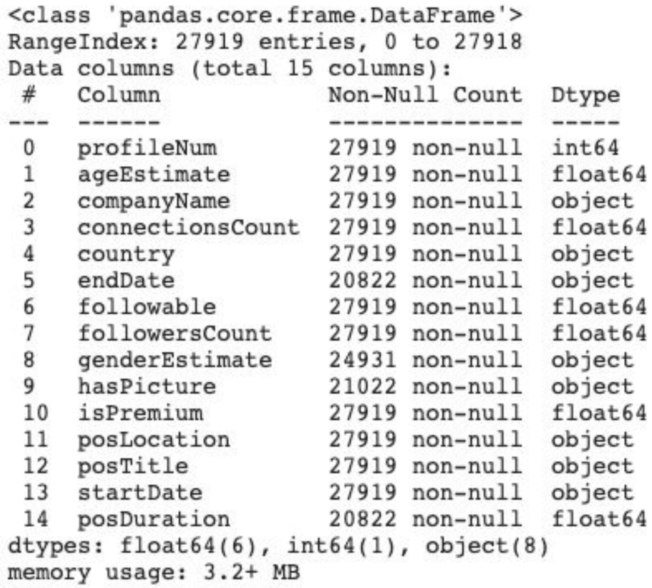


	profileNum	ageEstimate	companyName	connectionsCount	country	endDate	followable	follo
27914	39532	46.0	National Australia Bank	362.0	au	2009-04-01	1.0	
27915	39533	46.0	National Australia Bank	362.0	au	2007-05-01	1.0	
27916	39534	46.0	National Australia Bank	362.0	au	2003-08-01	1.0	
27917	39535	46.0	National Australia Bank	362.0	au	2000-06-01	1.0	
27918	39536	46.0	National Australia Bank	362.0	au	2000-06-01	1.0	

By default we get the last 5 rows

info()

- Return **all of the column names and its types**
- Get an idea of what the dataframe is like



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27919 entries, 0 to 27918
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   profileNum        27919 non-null  int64
 1   ageEstimate       27919 non-null  float64
 2   companyName       27919 non-null  object
 3   connectionsCount  27919 non-null  float64
 4   country           27919 non-null  object
 5   endDate           20822 non-null  object
 6   followable        27919 non-null  float64
 7   followersCount    27919 non-null  float64
 8   genderEstimate    24931 non-null  object
 9   hasPicture        21022 non-null  object
 10  isPremium         27919 non-null  float64
 11  posLocation       27919 non-null  object
 12  posTitle          27919 non-null  object
 13  startDate         27919 non-null  object
 14  posDuration       20822 non-null  float64
dtypes: float64(6), int64(1), object(8)
memory usage: 3.2+ MB
```

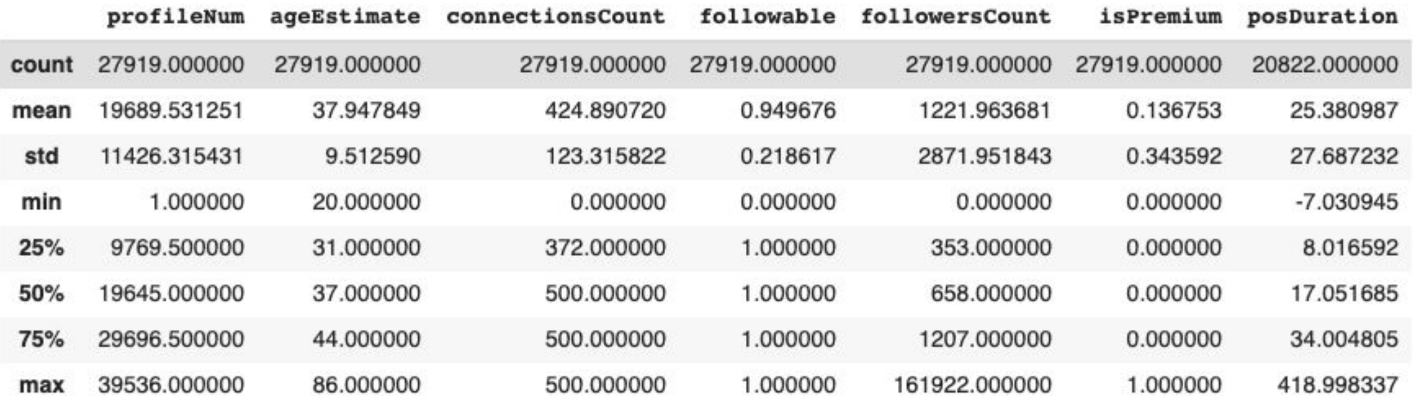
How many unique types are in this dataset?

Is the dataset uneven? If so list down the column with the most missing rows? (ie the most NULL rows)

Bit Project

describe()

- View summary statistics of numeric columns
- Provides a general idea of the dataset



	profileNum	ageEstimate	connectionsCount	followable	followersCount	isPremium	posDuration
count	27919.000000	27919.000000	27919.000000	27919.000000	27919.000000	27919.000000	20822.000000
mean	19689.531251	37.947849	424.890720	0.949676	1221.963681	0.136753	25.380987
std	11426.315431	9.512590	123.315822	0.218617	2871.951843	0.343592	27.687232
min	1.000000	20.000000	0.000000	0.000000	0.000000	0.000000	-7.030945
25%	9769.500000	31.000000	372.000000	1.000000	353.000000	0.000000	8.016592
50%	19645.000000	37.000000	500.000000	1.000000	658.000000	0.000000	17.051685
75%	29696.500000	44.000000	500.000000	1.000000	1207.000000	0.000000	34.004805
max	39536.000000	86.000000	500.000000	1.000000	161922.000000	1.000000	418.998337

What do you notice?

Observations

- Why is **describe** showing only 7 columns? Is it because of their **types** e.g(int,float,object)**?

- Notice the **statistics** for count, mean, std, min, etc, why isn’t the **count value** the same across all columns?

- In which **columns** do we care about the mean, standard deviation,..., max?

Bit Project

shape()

- Size of the dataset
- Returns the **number of rows and columns** in **(#rows, #columns)** format

Bit Project

Active Job Positions

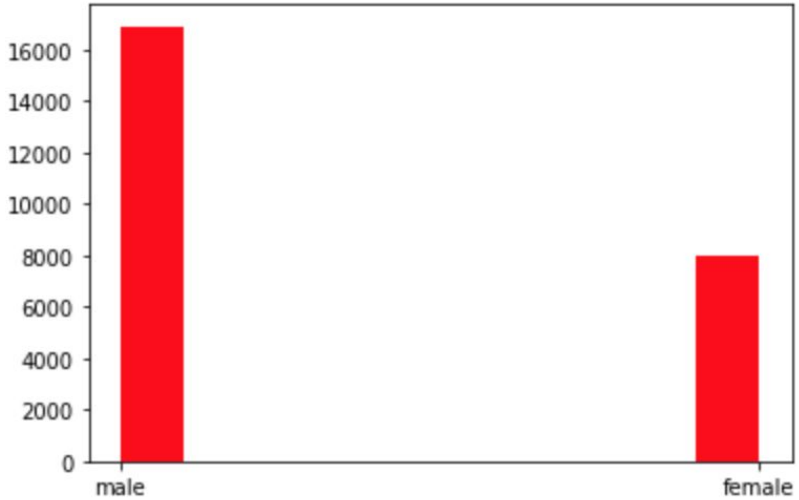
- **Goal**: count the number of active job positions
- **Method:** observe the endData
  - Run two functions:
    1) **isnull()** - check if the column has null values
    2) **sum()** - count the number of null values in the column

Completed Job Positions

- **Goal**: count the number of complete jobs
- **Method:** observe the endDate
    - Run two functions:
        1) **notnull()** - check if valid date
        2) **sum()** - count the number of null values in the column

Visualize the Gender Representation

Plot the data as a histogram looking at genederEstimate



HINT: linkedin_profiles['_______________'].hist(color='___')

Part 2
Getting Started With Data Wrangling

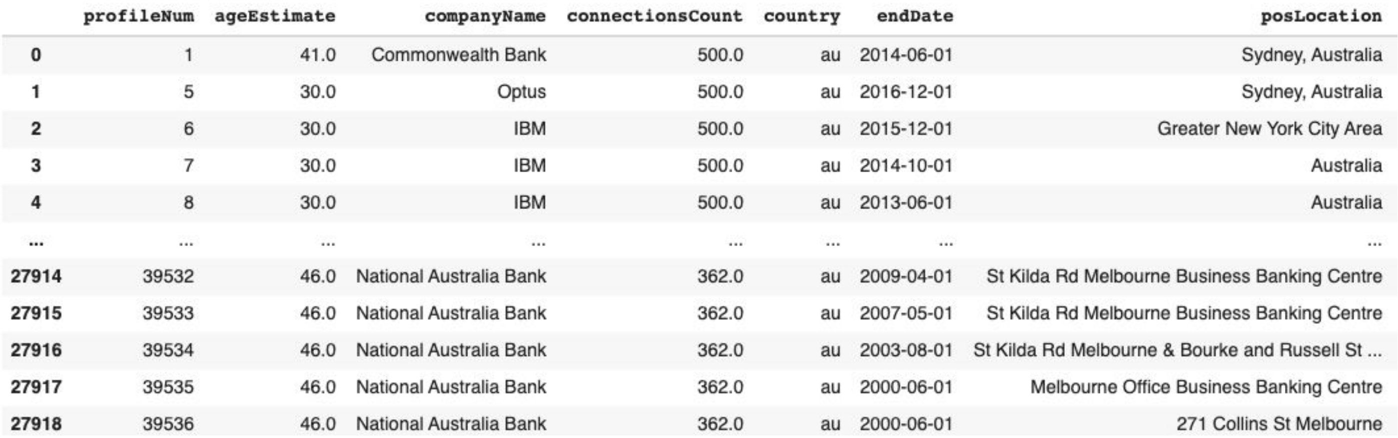
Getting Started with Data Wrangling

Now that we have observed the basic features of our dataset raw, we will began **cleaning** it.

Create a copy of the Original Dataset (Hint: use **.copy()**)

drop()

- Removes unneeded columns
- Use lists to help remove selected columns

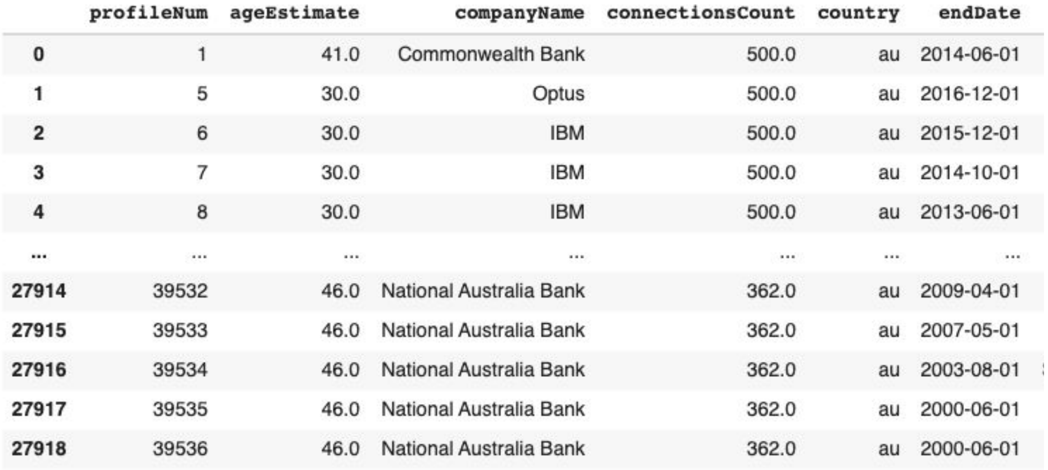


	profileNum	ageEstimate	companyName	connectionsCount	country	endDate	posLocation
0	1	41.0	Commonwealth Bank	500.0	au	2014-06-01	Sydney, Australia
1	5	30.0	Optus	500.0	au	2016-12-01	Sydney, Australia
2	6	30.0	IBM	500.0	au	2015-12-01	Greater New York City Area
3	7	30.0	IBM	500.0	au	2014-10-01	Australia
4	8	30.0	IBM	500.0	au	2013-06-01	Australia
...	...	...	...	...	...	...	...
27914	39532	46.0	National Australia Bank	362.0	au	2009-04-01	St Kilda Rd Melbourne Business Banking Centre
27915	39533	46.0	National Australia Bank	362.0	au	2007-05-01	St Kilda Rd Melbourne Business Banking Centre
27916	39534	46.0	National Australia Bank	362.0	au	2003-08-01	St Kilda Rd Melbourne & Bourke and Russell St ...
27917	39535	46.0	National Australia Bank	362.0	au	2000-06-01	Melbourne Office Business Banking Centre
27918	39536	46.0	National Australia Bank	362.0	au	2000-06-01	271 Collins St Melbourne

We removed columns = ['followable', 'followersCount', 'hasPicture', 'isPremium', 'genderEstimate']

dropna()

Drop cells with a **null or undefined value (NaN)**



	profileNum	ageEstimate	companyName	connectionsCount	country	endDate
0	1	41.0	Commonwealth Bank	500.0	au	2014-06-01
1	5	30.0	Optus	500.0	au	2016-12-01
2	6	30.0	IBM	500.0	au	2015-12-01
3	7	30.0	IBM	500.0	au	2014-10-01
4	8	30.0	IBM	500.0	au	2013-06-01
...	...	...	...	...	...	...
27914	39532	46.0	National Australia Bank	362.0	au	2009-04-01
27915	39533	46.0	National Australia Bank	362.0	au	2007-05-01
27916	39534	46.0	National Australia Bank	362.0	au	2003-08-01
27917	39535	46.0	National Australia Bank	362.0	au	2000-06-01
27918	39536	46.0	National Australia Bank	362.0	au	2000-06-01

We only want complete profiles. There are no null/invalid values!

Changing Column Names

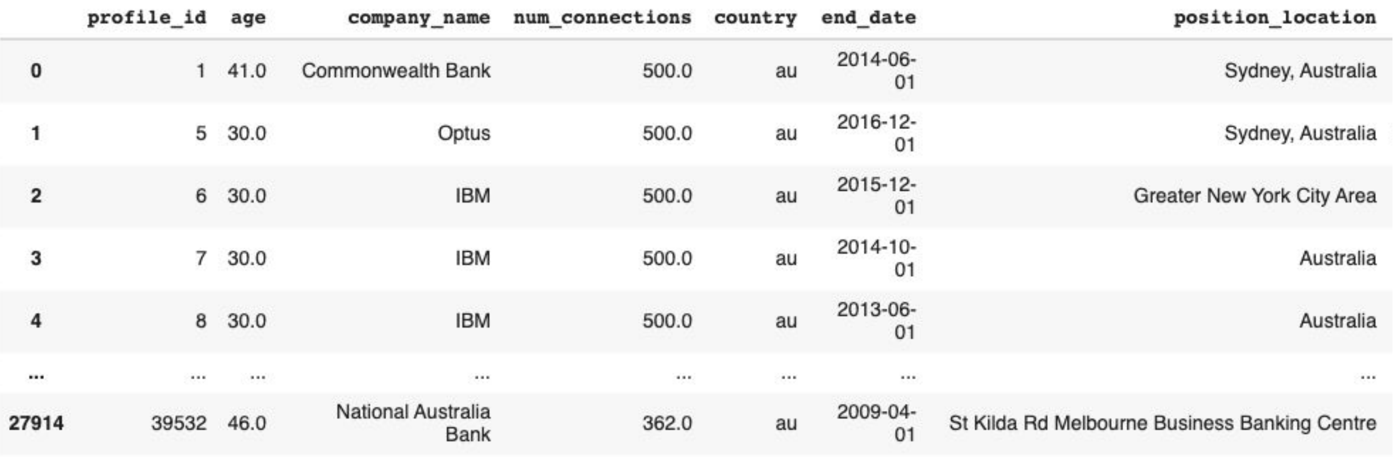
We do this for **readability.**

1) **List** the columns (Hint: use **columns**)
2) Rename the columns to be in **snakecase** form (lowercase words separated by an underscore)

    e.g. my_sample_variable_name.

The column names are: [(profileId), ageEstimate, companyName, country, endDate, genderEstimate, startDate]

Changing Column Names

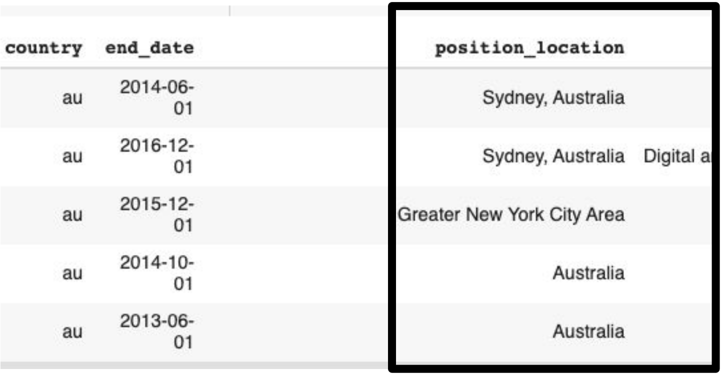


	profile_id	age	company_name	num_connections	country	end_date	position_location
0	1	41.0	Commonwealth Bank	500.0	au	2014-06-01	Sydney, Australia
1	5	30.0	Optus	500.0	au	2016-12-01	Sydney, Australia
2	6	30.0	IBM	500.0	au	2015-12-01	Greater New York City Area
3	7	30.0	IBM	500.0	au	2014-10-01	Australia
4	8	30.0	IBM	500.0	au	2013-06-01	Australia
...	...	...	...	...	...	...	...
27914	39532	46.0	National Australia Bank	362.0	au	2009-04-01	St Kilda Rd Melbourne Business Banking Centre

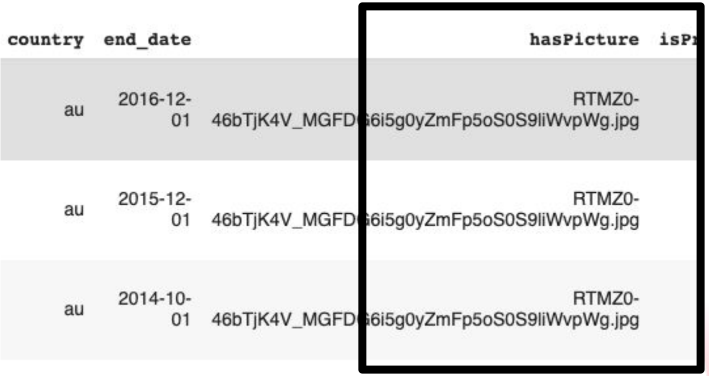
We should get this df

reindex()

- Changes the **order** of columns



country	end_date	position_location	
au	2014-06-01	Sydney, Australia	
au	2016-12-01	Sydney, Australia	Digital a
au	2015-12-01	Greater New York City Area	
au	2014-10-01	Australia	
au	2013-06-01	Australia	

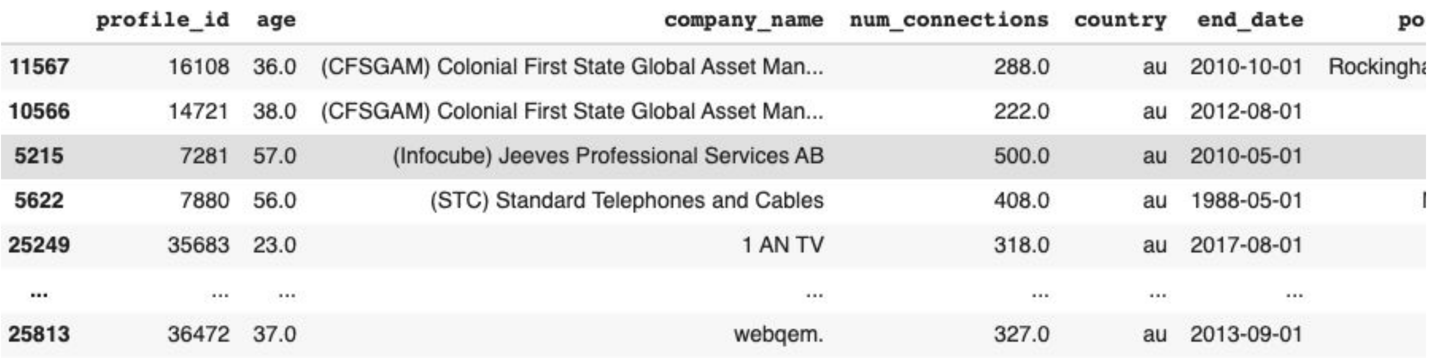


country	end_date	hasPicture	isPr
au	2016-12-01	RTMZ0-46bTjK4V_MGFDG6i5g0yZmFp5oS0S9liWvpWg.jpg	
au	2015-12-01	RTMZ0-46bTjK4V_MGFDG6i5g0yZmFp5oS0S9liWvpWg.jpg	
au	2014-10-01	RTMZ0-46bTjK4V_MGFDG6i5g0yZmFp5oS0S9liWvpWg.jpg	

Notice how the columns have changed?

sort_values()

sort_values(by = ‘selection’) sorts the table values according to the selection



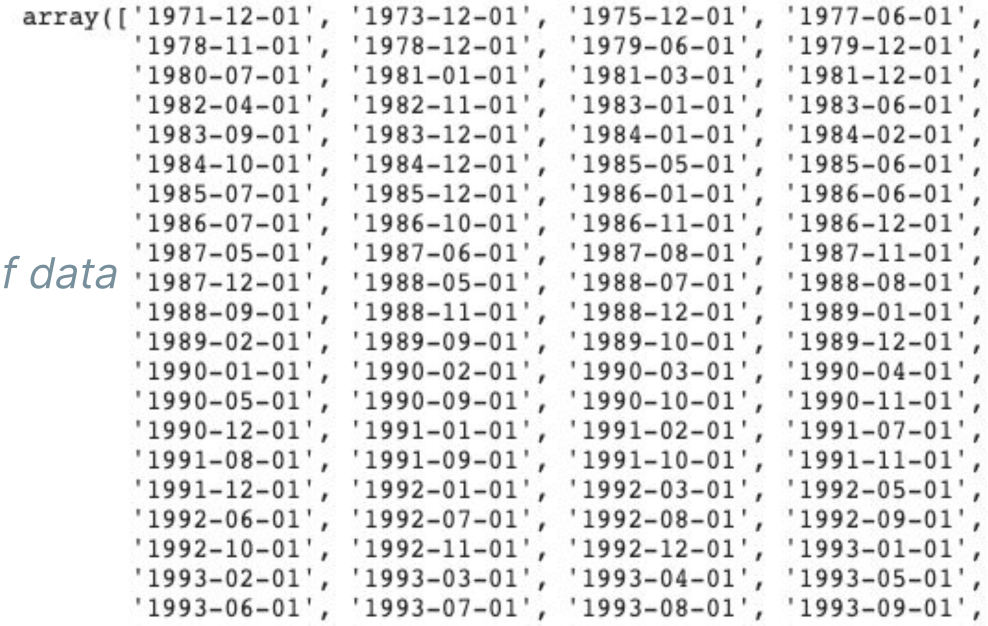
	profile_id	age	company_name	num_connections	country	end_date	po
11567	16108	36.0	(CFSGAM) Colonial First State Global Asset Man...	288.0	au	2010-10-01	Rockingha
10566	14721	38.0	(CFSGAM) Colonial First State Global Asset Man...	222.0	au	2012-08-01	
5215	7281	57.0	(Infocube) Jeeves Professional Services AB	500.0	au	2010-05-01	
5622	7880	56.0	(STC) Standard Telephones and Cables	408.0	au	1988-05-01	
25249	35683	23.0	1 AN TV	318.0	au	2017-08-01	
...	...	...	...	...	...	...	
25813	36472	37.0	webqem.	327.0	au	2013-09-01	

We are sorted by company name in ascending order

Finding Unique and Similar Strings

First, list down all the **unique names** in these columns.
Next, sort these in **alphabetical** order.

WOAH! That’s a lot of data



```
array(['1971-12-01', '1973-12-01', '1975-12-01', '1977-06-01',
       '1978-11-01', '1978-12-01', '1979-06-01', '1979-12-01',
       '1980-07-01', '1981-01-01', '1981-03-01', '1981-12-01',
       '1982-04-01', '1982-11-01', '1983-01-01', '1983-06-01',
       '1983-09-01', '1983-12-01', '1984-01-01', '1984-02-01',
       '1984-10-01', '1984-12-01', '1985-05-01', '1985-06-01',
       '1985-07-01', '1985-12-01', '1986-01-01', '1986-06-01',
       '1986-07-01', '1986-10-01', '1986-11-01', '1986-12-01',
       '1987-05-01', '1987-06-01', '1987-08-01', '1987-11-01',
       '1987-12-01', '1988-05-01', '1988-07-01', '1988-08-01',
       '1988-09-01', '1988-11-01', '1988-12-01', '1989-01-01',
       '1989-02-01', '1989-09-01', '1989-10-01', '1989-12-01',
       '1990-01-01', '1990-02-01', '1990-03-01', '1990-04-01',
       '1990-05-01', '1990-09-01', '1990-10-01', '1990-11-01',
       '1990-12-01', '1991-01-01', '1991-02-01', '1991-07-01',
       '1991-08-01', '1991-09-01', '1991-10-01', '1991-11-01',
       '1991-12-01', '1992-01-01', '1992-03-01', '1992-05-01',
       '1992-06-01', '1992-07-01', '1992-08-01', '1992-09-01',
       '1992-10-01', '1992-11-01', '1992-12-01', '1993-01-01',
       '1993-02-01', '1993-03-01', '1993-04-01', '1993-05-01',
       '1993-06-01', '1993-07-01', '1993-08-01', '1993-09-01',
```

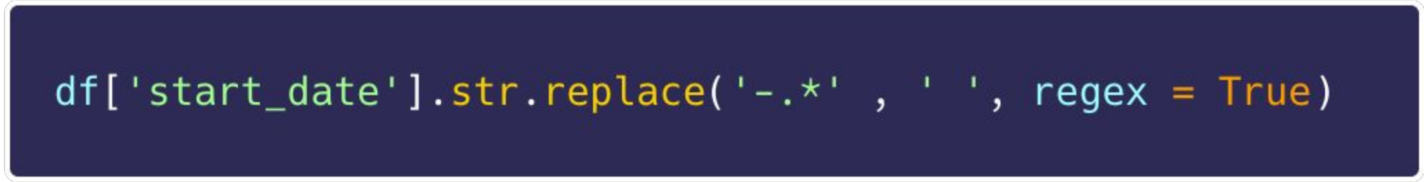
df[‘column_name’].str.replace()

df['columun_name'].str.replace('string to find', 'string to replace') replaces unwanted parts of a string. This command **finds** the string we have specified and **replaces** it with what we want.

In this case, we will be using **Regular Expression Matching** to search for **patterns** of a string that follow specific rules.

Regular Expression Matching

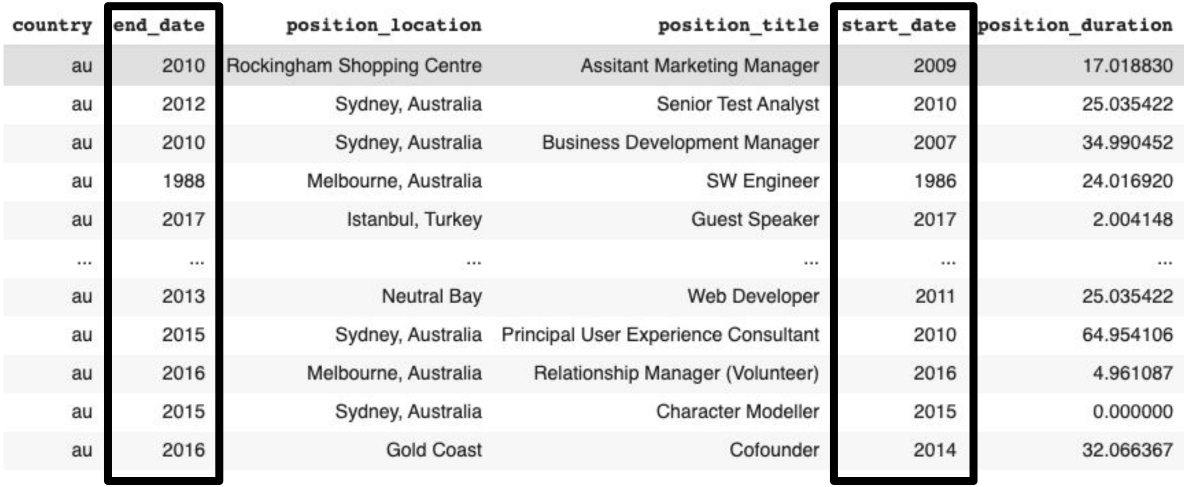
- **Problem:** Change start_date and end_date format
- **Find:** YEAR-MONTH-DAY
- **Replace:** YEAR
- **Pattern:** -.*
  - .* substitutes for any string of characters
  - remove by replace the pattern with an empty string



```
df['start_date'].str.replace('-.*' , ' ', regex = True)
```

Regular Expression Matching

We should get something like this



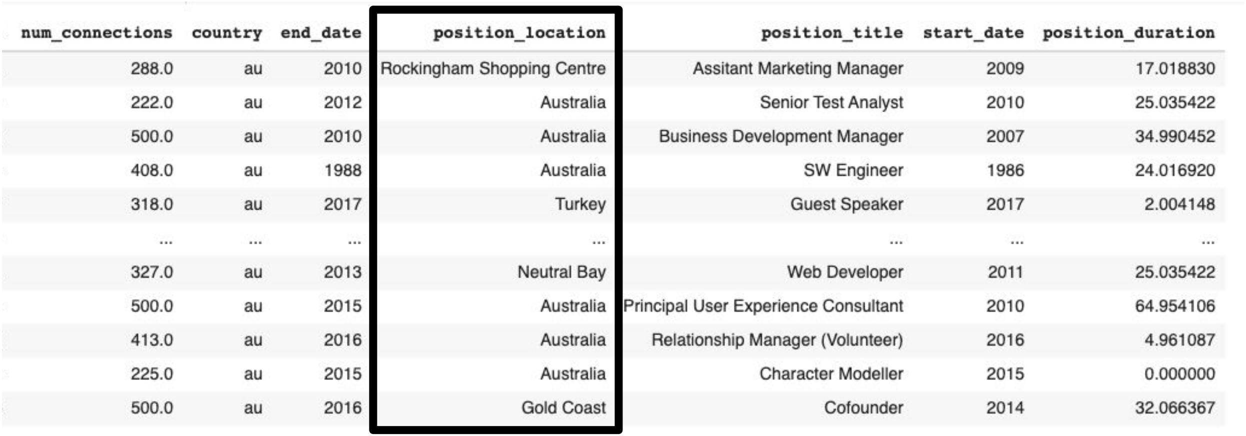
country	end_date	position_location	position_title	start_date	position_duration
au	2010	Rockingham Shopping Centre	Assitant Marketing Manager	2009	17.018830
au	2012	Sydney, Australia	Senior Test Analyst	2010	25.035422
au	2010	Sydney, Australia	Business Development Manager	2007	34.990452
au	1988	Melbourne, Australia	SW Engineer	1986	24.016920
au	2017	Istanbul, Turkey	Guest Speaker	2017	2.004148
...	...	...	...	...	...
au	2013	Neutral Bay	Web Developer	2011	25.035422
au	2015	Sydney, Australia	Principal User Experience Consultant	2010	64.954106
au	2016	Melbourne, Australia	Relationship Manager (Volunteer)	2016	4.961087
au	2015	Sydney, Australia	Character Modeller	2015	0.000000
au	2016	Gold Coast	Cofounder	2014	32.066367

Regular Expression Matching

- **Problem:** Change 'position_location' column format
- **Find:** Format of city, country
- **Replace:** Remove city
- **Pattern:** .*,
  - .* substitutes for any string of characters
  - remove by replacing everything before ,

Regular Expression Matching

We should get something like this

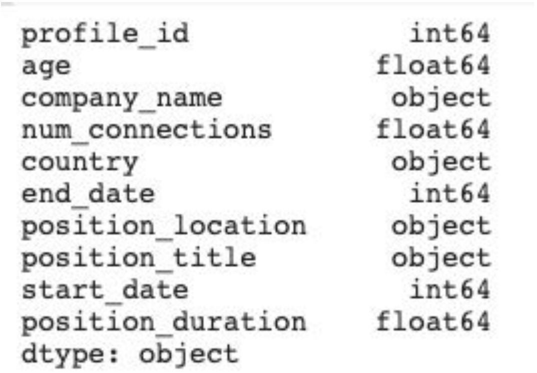


num_connections	country	end_date	position_location	position_title	start_date	position_duration
288.0	au	2010	Rockingham Shopping Centre	Assitant Marketing Manager	2009	17.018830
222.0	au	2012	Australia	Senior Test Analyst	2010	25.035422
500.0	au	2010	Australia	Business Development Manager	2007	34.990452
408.0	au	1988	Australia	SW Engineer	1986	24.016920
318.0	au	2017	Turkey	Guest Speaker	2017	2.004148
...	...	...	...	...	...	...
327.0	au	2013	Neutral Bay	Web Developer	2011	25.035422
500.0	au	2015	Australia	Principal User Experience Consultant	2010	64.954106
413.0	au	2016	Australia	Relationship Manager (Volunteer)	2016	4.961087
225.0	au	2015	Australia	Character Modeller	2015	0.000000
500.0	au	2016	Gold Coast	Cofounder	2014	32.066367

Part 3
Micro Wrangling and Visualization

Micro Wrangling and Visualization

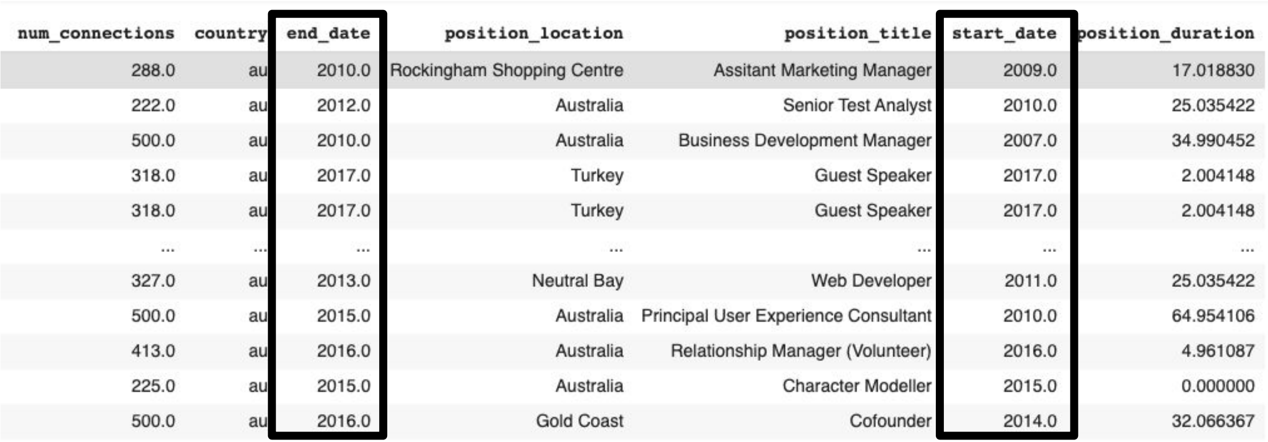
- Dividing our dataset into **multiple smaller dataframes**
- We will base dataframes on the **start_date data** for a job position
  - **3 intervals:**
    - 1960 to 1979
    - 1980 to 1999
    - 2000 to 2019



```
profile_id                int64
age                     float64
company_name             object
num_connections         float64
country                  object
end_date                  int64
position_location        object
position_title           object
start_date                int64
position_duration       float64
dtype: object
```

HINT: We might want to convert start_date and end_date values to integers

Between 2000 to 2019



num_connections	country	end_date	position_location	position_title	start_date	position_duration
288.0	au	2010.0	Rockingham Shopping Centre	Assitant Marketing Manager	2009.0	17.018830
222.0	au	2012.0	Australia	Senior Test Analyst	2010.0	25.035422
500.0	au	2010.0	Australia	Business Development Manager	2007.0	34.990452
318.0	au	2017.0	Turkey	Guest Speaker	2017.0	2.004148
318.0	au	2017.0	Turkey	Guest Speaker	2017.0	2.004148
...	...	...	...	...	...	...
327.0	au	2013.0	Neutral Bay	Web Developer	2011.0	25.035422
500.0	au	2015.0	Australia	Principal User Experience Consultant	2010.0	64.954106
413.0	au	2016.0	Australia	Relationship Manager (Volunteer)	2016.0	4.961087
225.0	au	2015.0	Australia	Character Modeller	2015.0	0.000000
500.0	au	2016.0	Gold Coast	Cofounder	2014.0	32.066367

How many profiles started their job between 2000 to 2019?

Hint: get the dimensions of the dataframe

Bit Project

Visualizing New Jobs During 2000-2019

Set 1: Graphing by the **company names**. See what companies hired new employees between 2000 to 2019.

But which graph should we use?

Choosing Graphs

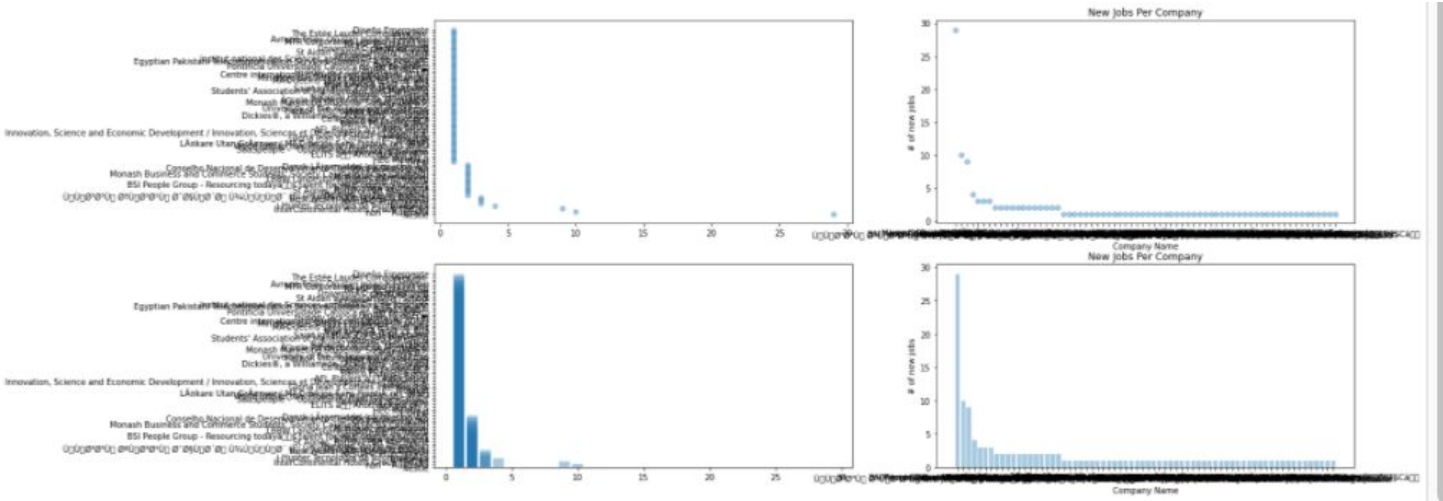
Which of these graphs seem useful and which ones are unnecessary?

Did the visualization style influence your decision?

Select the two graphs you think are more useful and add a **title** for both subplots, **xlabel**, and **ylabel**.

Choosing Graphs

What kind of graph is this? Would this be helpful?



Thank You!