In this section of jupyter notebook, I will work on levels_Fyi_salary_df. Lets start the data wrangling with this dataframe.

In [2]:
```
import pandas as pd
```

In [4]:
```
### First we need to load the data
levels_Fyi_salary_df = pd.read_csv("/Users/ertuboston/Documents/Data_Science_Merrimack/DSE500
```

In [5]:
```
levels_Fyi_salary_df.head(6)
```

Out[5]:

	timestamp	company	level	title	totalyearlycompensation	location	yearsofexperience	yearsa
0	6/7/2017 11:33:27	Oracle	L3	Product Manager	127000	Redwood City, CA	1.5	
1	6/10/2017 17:11:29	eBay	SE 2	Software Engineer	100000	San Francisco, CA	5.0	
2	6/11/2017 14:53:57	Amazon	L7	Product Manager	310000	Seattle, WA	8.0	
3	6/17/2017 0:23:14	Apple	M1	Software Engineering Manager	372000	Sunnyvale, CA	7.0	
4	6/20/2017 10:58:51	Microsoft	60	Software Engineer	157000	Mountain View, CA	5.0	
5	6/21/2017 17:27:47	Microsoft	63	Software Engineer	208000	Seattle, WA	8.5	

6 rows × 29 columns

In [6]:
```
### Lets check the sum of missing values in each column in levels_Fyi_salary_df
levels_Fyi_salary_df.isna().sum()
```

Out[6]:
```
timestamp                    0
company                      5
level                      123
title                        0
totalyearlycompensation      0
location                     0
yearsofexperience            0
yearsatcompany               0
tag                        870
basesalary                   0
stockgrantvalue              0
bonus                        0
gender                   19540
otherdetails             22508
cityid                       0
dmaid                        2
rowNumber                    0
Masters_Degree               0
Bachelors_Degree             0
Doctorate_Degree             0
Highschool                   0
Some_College                 0
Race_Asian                   0
Race_White                   0
Race_Two_Or_More             0
Race_Black                   0
Race_Hispanic                0
Race                     40215
Education                32272
dtype: int64
```

In [7]:
```
### lets see it as percentage
missing_values_percentages = levels_Fyi_salary_df.isna().sum() / len(levels_Fyi_salary_df) *1
```

```
missing_values_percentages
### As we see that columns gender,other details,
### Race and education columns have a good amount of missing values
```

Out[7]:
```
timestamp                  0.000000
company                    0.007982
level                      0.196354
title                      0.000000
totalyearlycompensation    0.000000
location                   0.000000
yearsofexperience          0.000000
yearsatcompany             0.000000
tag                        1.388845
basesalary                 0.000000
stockgrantvalue            0.000000
bonus                      0.000000
gender                    31.193129
otherdetails              35.931164
cityid                     0.000000
dmaid                      0.003193
rowNumber                  0.000000
Masters_Degree             0.000000
Bachelors_Degree           0.000000
Doctorate_Degree           0.000000
Highschool                 0.000000
Some_College               0.000000
Race_Asian                 0.000000
Race_White                 0.000000
Race_Two_Or_More           0.000000
Race_Black                 0.000000
Race_Hispanic              0.000000
Race                      64.198142
Education                 51.518151
dtype: float64
```

In [8]:
```
### we could switch the na s with something else or default
### but since the columns that have missing values, will not affect our search.
```

```
### we drop the NA values from the dataframe.

levels_Fyi_salary_df_sample = levels_Fyi_salary_df.dropna()
levels_Fyi_salary_df_sample
```

Out[8]:

	timestamp	company	level	title	totalyearlycompensation	location	yearsofexperience	y
15710	1/27/2020 22:59:06	Google	L6	Software Engineer	400000	Sunnyvale, CA	5.0	
23532	7/3/2020 19:56:38	Microsoft	61	Software Engineer	136000	Redmond, WA	3.0	
23533	7/3/2020 20:03:57	Google	L5	Software Engineer	337000	San Bruno, CA	6.0	
23534	7/3/2020 20:05:37	Microsoft	62	Software Engineer	222000	Seattle, WA	4.0	
23535	7/3/2020 20:19:06	Blend	IC3	Software Engineer	187000	San Francisco, CA	5.0	
...	...	...	...	...	...	...	...	
61981	2/15/2021 19:50:36	Facebook	M2	Software Engineering Manager	1470000	Menlo Park, CA	9.0	
61982	3/9/2021 17:03:07	Google	L10	Product Manager	4500000	San Francisco, CA	20.0	
61984	3/25/2021 10:45:03	Zapier	L8	Software Engineering Manager	1605000	Denver, CO	16.0	
61987	5/18/2021 15:34:21	Facebook	D1	Software Engineering	2372000	Menlo Park, CA	22.0	

	timestamp	company	level	title	totalyearlycompensation	location	yearsofexperience	y
				Manager				
61991	7/30/2021 22:23:24	Facebook	E9	Product Manager	4980000	Menlo Park, CA	17.0	

21515 rows × 29 columns

In [11]:
```
### let's see if it gives us enough data to work on
missing_values_counts = levels_Fyi_salary_df_sample.isna().sum()
missing_values_counts

### As we see there are no missing values anymore
### our data frame went down from 62643 columns to 21515.
### Still we have a good size of data to work on.
```

Out[11]:
```
timestamp                0
company                  0
level                    0
title                    0
totalyearlycompensation  0
location                 0
yearsofexperience        0
yearsatcompany           0
tag                      0
basesalary               0
stockgrantvalue          0
bonus                    0
gender                   0
otherdetails             0
cityid                   0
dmaid                    0
rowNumber                0
Masters_Degree           0
Bachelors_Degree         0
Doctorate_Degree         0
Highschool               0
Some_College             0
Race_Asian               0
Race_White               0
Race_Two_Or_More         0
Race_Black               0
Race_Hispanic            0
Race                     0
Education                0
dtype: int64
```

==========

After dealing with the missing values in leves_Fyi_salary dataframe, now there are some columns that we won't need it to do our research, such as race, gender, etc... we can remove those columns from levels_Fyi_salary_df_sample and focus on only the columns we would need.

In [17]:
```
print(levels_Fyi_salary_df_sample.columns)
```

```
Index(['timestamp', 'company', 'level', 'title', 'totalyearlycompensation',
       'location', 'yearsofexperience', 'yearsatcompany', 'tag', 'basesalary',
       'stockgrantvalue', 'bonus', 'gender', 'otherdetails', 'cityid', 'dmaid',
       'rowNumber', 'Masters_Degree', 'Bachelors_Degree', 'Doctorate_Degree',
       'Highschool', 'Some_College', 'Race_Asian', 'Race_White',
       'Race_Two_Or_More', 'Race_Black', 'Race_Hispanic', 'Race', 'Education'],
      dtype='object')
```

In [19]:
```
levels_Fyi_salary_df_sample.drop(columns=['gender', 'otherdetails','tag','yearsofexperience',
                                          'yearsatcompany',
                                          'rowNumber', 'Masters_Degree', 'Bachelors_Degree',
                                          'Doctorate_Degree','Highschool', 'Some_College',
                                          'Race_Asian', 'Race_White', 'Race_Two_Or_More', 'Race
                                          'Race_Hispanic', 'Race', 'Education'], inplace = True
levels_Fyi_salary_df_sample

### Now we have 11 columns instead of 29 columns
```

```
/var/folders/yn/lfh7s3f52q18zdwkxgbxg58r0000gn/T/ipykernel_7294/325370154.py:1: SettingWithCop
yWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/
indexing.html#returning-a-view-versus-a-copy
  levels_Fyi_salary_df_sample.drop(columns=['gender', 'otherdetails','tag','yearsofexperienc
e',
```

Out[19]:

	timestamp	company	level	title	totalyearlycompensation	location	basesalary	stockgra
15710	1/27/2020 22:59:06	Google	L6	Software Engineer	400000	Sunnyvale, CA	210000.0	1
23532	7/3/2020 19:56:38	Microsoft	61	Software Engineer	136000	Redmond, WA	124000.0	
23533	7/3/2020 20:03:57	Google	L5	Software Engineer	337000	San Bruno, CA	177000.0	1
23534	7/3/2020 20:05:37	Microsoft	62	Software Engineer	222000	Seattle, WA	164000.0	
23535	7/3/2020 20:19:06	Blend	IC3	Software Engineer	187000	San Francisco, CA	165000.0	
...	...	...	...	...	...	...	...	
61981	2/15/2021 19:50:36	Facebook	M2	Software Engineering Manager	1470000	Menlo Park, CA	290000.0	
61982	3/9/2021 17:03:07	Google	L10	Product Manager	4500000	San Francisco, CA	450000.0	
61984	3/25/2021 10:45:03	Zapier	L8	Software Engineering Manager	1605000	Denver, CO	250000.0	
61987	5/18/2021 15:34:21	Facebook	D1	Software Engineering Manager	2372000	Menlo Park, CA	315000.0	

	timestamp	company	level	title	totalyearlycompensation	location	basesalary	stockgra
61991	7/30/2021 22:23:24	Facebook	E9	Product Manager	4980000	Menlo Park, CA	380000.0	

21515 rows × 11 columns

In [59]:
```
### Now I would like to create a dataframe where title is only Data Scientist
### I use str.contains, because I would like to get data scientist, lead data scientist and e
### Our dataset went down to 872 rows and 11 columns

levels_salary_DS = levels_Fyi_salary_df_sample[levels_Fyi_salary_df_sample['title']
.str.contains('Data Scientist', case = False) ]

levels_salary_DS
```

Out[59]:

	timestamp	company	level	title	totalyearlycompensation	location	basesalary	stockg
23679	7/6/2020 17:16:12	Google	L3	Data Scientist	170000	San Francisco, CA	170000.0	
23685	7/6/2020 18:03:05	Facebook	IC4	Data Scientist	205000	Menlo Park, CA	150000.0	
23699	7/6/2020 22:10:39	Microsoft	62	Data Scientist	220000	Bellevue, WA	150000.0	
23702	7/6/2020 22:31:17	PayPal	T24	Data Scientist	216000	San Jose, CA	160000.0	
23724	7/7/2020 8:03:56	Amazon	Senior	Data Scientist	185000	Cambridge, MA	185000.0	
...	...	...	...	...	...	...	...	
61592	8/14/2021 23:00:24	Netflix	Senior Data Scientist	Data Scientist	605000	Los Gatos, CA	605000.0	
61642	8/15/2021 12:58:07	Facebook	IC4	Data Scientist	185000	Tel Aviv, TA, Israel	133000.0	
61687	8/15/2021 22:31:26	Adobe	L3	Data Scientist	250000	San Jose, CA	150000.0	
61793	8/16/2021 21:02:37	Xandr	L1	Data Scientist	120000	Portland, OR	110000.0	
61803	8/16/2021 22:19:48	Facebook	L4	Data Scientist	233000	Menlo Park, CA	157000.0	

872 rows × 11 columns

In [23]:
```
### Since timestamp column's type is object, lets convert it to date format.
print(levels_salary_DS.dtypes)
```

```
timestamp                 object
company                   object
level                     object
title                     object
totalyearlycompensation    int64
location                  object
basesalary               float64
stockgrantvalue          float64
bonus                    float64
cityid                     int64
dmaid                    float64
dtype: object
```

In [25]:
```
### timestamp type was object and converted to datetime
### in any case, we might need it

levels_salary_DS['timestamp'] = pd.to_datetime(levels_salary_DS['timestamp'])
levels_salary_DS['timestamp']
```

```
/var/folders/yn/lfh7s3f52q18zdwkxgbxg58r0000gn/T/ipykernel_7294/2065285718.py:4: SettingWithCo
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/
indexing.html#returning-a-view-versus-a-copy
  levels_salary_DS['timestamp'] = pd.to_datetime(levels_salary_DS['timestamp'])
```

Out[25]:
```
23679   2020-07-06 17:16:12
23685   2020-07-06 18:03:05
23699   2020-07-06 22:10:39
23702   2020-07-06 22:31:17
23724   2020-07-07 08:03:56
                ...
61592   2021-08-14 23:00:24
61642   2021-08-15 12:58:07
61687   2021-08-15 22:31:26
61793   2021-08-16 21:02:37
61803   2021-08-16 22:19:48
Name: timestamp, Length: 872, dtype: datetime64[ns]
```

In [27]:
```
print(levels_salary_DS.columns)
```

```
Index(['timestamp', 'company', 'level', 'title', 'totalyearlycompensation',
       'location', 'basesalary', 'stockgrantvalue', 'bonus', 'cityid',
       'dmaid'],
      dtype='object')
```

In [29]:
```
levels_salary_DS
```

Out[29]:

	timestamp	company	level	title	totalyearlycompensation	location	basesalary	stockg
23679	2020-07-06 17:16:12	Google	L3	Data Scientist	170000	San Francisco, CA	170000.0	
23685	2020-07-06 18:03:05	Facebook	IC4	Data Scientist	205000	Menlo Park, CA	150000.0	
23699	2020-07-06 22:10:39	Microsoft	62	Data Scientist	220000	Bellevue, WA	150000.0	
23702	2020-07-06 22:31:17	PayPal	T24	Data Scientist	216000	San Jose, CA	160000.0	
23724	2020-07-07 08:03:56	Amazon	Senior	Data Scientist	185000	Cambridge, MA	185000.0	
...	...	...	...	...	...	...	...	
61592	2021-08-14 23:00:24	Netflix	Senior Data Scientist	Data Scientist	605000	Los Gatos, CA	605000.0	
61642	2021-08-15 12:58:07	Facebook	IC4	Data Scientist	185000	Tel Aviv, TA, Israel	133000.0	
61687	2021-08-15 22:31:26	Adobe	L3	Data Scientist	250000	San Jose, CA	150000.0	

	timestamp	company	level	title	totalyearlycompensation	location	basesalary	stockgr
61793	2021-08-16 21:02:37	Xandr	L1	Data Scientist	120000	Portland, OR	110000.0	
61803	2021-08-16 22:19:48	Facebook	L4	Data Scientist	233000	Menlo Park, CA	157000.0	

872 rows × 11 columns

In [31]:
```
levels_salary_DS['location'].unique()
### as we see here in location column there are city, state and country combination
### We can put them in separate columns as city, state, country
```

Out[31]:
```
array(['San Francisco, CA', 'Menlo Park, CA', 'Bellevue, WA',
       'San Jose, CA', 'Cambridge, MA', 'Dallas, TX', 'Hillsboro, OR',
       'Seattle, WA', 'London, EN, United Kingdom', 'Redmond, WA',
       'Cupertino, CA', 'Sunnyvale, CA', 'Richmond, VA', 'Italy, TX',
       'New York, NY', 'Singapore, SG, Singapore', 'Austin, TX',
       'Boston, MA', 'Moscow, MC, Russia', 'Redwood City, CA',
       'Arizona City, AZ', 'San Diego, CA', 'Pleasanton, CA',
       'Bentonville, AR', 'Bangalore, KA, India', 'Palo Alto, CA',
       'Los Gatos, CA', 'Los Angeles, CA', 'Washington, DC',
       'Mountain View, CA', 'Bridgewater, NJ', 'Columbus, OH',
       'Santa Clara, CA', 'Mumbai, MH, India', 'Atlanta, GA',
       'Antioch, TN', 'Cleveland, OH', 'Portland, OR',
       'Hyderabad, TS, India', 'Kansas City, KS', 'Chicago, IL',
       'Charlotte, NC', 'Berlin, BE, Germany', 'Armonk, NY',
       'Bengaluru, KA, India', 'Shelton, CT', 'Tel Aviv, TA, Israel',
       'Zurich, ZH, Switzerland', 'Newtown Square, PA', 'Fairfax, VA',
       'Irvine, CA', 'San Antonio, TX', 'Toronto, ON, Canada',
       'Kirkland, WA', 'Raleigh, NC', 'Munich, BY, Germany',
       'Louisville, KY', 'Spring, TX', 'Provo, UT', 'Fremont, CA',
       'Basel, BS, Switzerland', 'Bloomington, IL', 'Beaverton, OR',
       'Houston, TX', 'Arlington, VA', 'Needham, MA',
       'Chennai, TN, India', 'Oakland, CA', 'Paris, IL, France',
       'Beijing, BJ, China', 'Dublin, DN, Ireland',
       'The Hague, ZH, Netherlands', 'Boulder, CO', 'Ottawa, ON, Canada',
       'Plano, TX', 'Mill Valley, CA', 'Gurgaon, HR, India',
       'Wilmington, DE', 'Amsterdam, NH, Netherlands',
       'Barcelona, CT, Spain', 'Reston, VA', 'Karlsruhe, BW, Germany',
       'Minneapolis, MN', 'Phoenix, AZ', 'Chandler, AZ',
       'Hyderabad, AP, India', 'Kiev, KC, Ukraine', 'Charleston, SC',
       'Melbourne, VI, Australia', 'Longmont, CO', 'Glendale, CA',
       'Columbia, MD', 'Traverse City, MI', 'Madison, WI',
       'Wellesley, MA', 'Las Vegas, NV', 'Jersey City, NJ',
       'Huntsville, AL', 'Luxembourg, LU, Luxembourg',
```

```
       'Saint Petersburg, SP, Russia', 'South San Francisco, CA',
       'Manassas, VA', 'Basking Ridge, NJ', 'Foster City, CA',
       'Rochester, MN', 'Cincinnati, OH', 'Saint Paul, MN',
       'Vienna, WI, Austria', 'Hoboken, NJ', 'Montreal, QC, Canada',
       'Birmingham, AL', 'Pittsburgh, PA', 'Detroit, MI', 'Vienna, VA',
       'Philadelphia, PA', 'Worcester, MA', 'Alpharetta, GA',
       'Santa Monica, CA', 'Dearborn, MI', 'Berkeley, CA',
       'Chicago Heights, IL', 'San Bruno, CA', 'Brooklyn, NY',
       'Newport Beach, CA', 'Canberra, CT, Australia', 'Orlando, FL',
       'Hartford, CT', 'Vancouver, BC, Canada', 'Tulsa, OK',
       'Mississippi State, MS', 'Miami, FL', 'Warsaw, MZ, Poland',
       'Stockholm, ST, Sweden', 'King of Prussia, PA', 'Indianapolis, IN',
       'Pune, MH, India', 'Memphis, TN', 'Milpitas, CA',
       'Sydney, NS, Australia', 'Boise, ID', 'Denver, CO',
       'St. Louis, MO', 'Taichung City, TP, Taiwan', 'Taipei, TP, Taiwan',
       'Washington, VA', 'Tokyo, TY, Japan', 'Annapolis Junction, MD'],
      dtype=object)
```

In [33]:

```
### We can split them in separate columns as city, state, country

location_split = levels_salary_DS['location'].str.split(', ', expand=True)

# Assign the split parts to new columns
levels_salary_DS['city'] = location_split[0]
levels_salary_DS['state'] = location_split[1]
levels_salary_DS['country'] = location_split[2]

levels_salary_DS
```

```
/var/folders/yn/lfh7s3f52q18zdwkxgbxg58r0000gn/T/ipykernel_7294/1285978410.py:6: SettingWithCo
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/
indexing.html#returning-a-view-versus-a-copy
  levels_salary_DS['city'] = location_split[0]
/var/folders/yn/lfh7s3f52q18zdwkxgbxg58r0000gn/T/ipykernel_7294/1285978410.py:7: SettingWithCo
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/
indexing.html#returning-a-view-versus-a-copy
  levels_salary_DS['state'] = location_split[1]
/var/folders/yn/lfh7s3f52q18zdwkxgbxg58r0000gn/T/ipykernel_7294/1285978410.py:8: SettingWithCo
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/
indexing.html#returning-a-view-versus-a-copy
  levels_salary_DS['country'] = location_split[2]
```

Out[33]:

	timestamp	company	level	title	totalyearlycompensation	location	basesalary	stockg
23679	2020-07-06 17:16:12	Google	L3	Data Scientist	170000	San Francisco, CA	170000.0	
23685	2020-07-06 18:03:05	Facebook	IC4	Data Scientist	205000	Menlo Park, CA	150000.0	
23699	2020-07-06 22:10:39	Microsoft	62	Data Scientist	220000	Bellevue, WA	150000.0	
23702	2020-07-06 22:31:17	PayPal	T24	Data Scientist	216000	San Jose, CA	160000.0	
23724	2020-07-07 08:03:56	Amazon	Senior	Data Scientist	185000	Cambridge, MA	185000.0	
...	...	...	...	...	...	...	...	
61592	2021-08-14 23:00:24	Netflix	Senior Data Scientist	Data Scientist	605000	Los Gatos, CA	605000.0	
61642	2021-08-15 12:58:07	Facebook	IC4	Data Scientist	185000	Tel Aviv, TA, Israel	133000.0	
61687	2021-08-15 22:31:26	Adobe	L3	Data Scientist	250000	San Jose, CA	150000.0	

	timestamp	company	level	title	totalyearlycompensation	location	basesalary	stockgr
61793	2021-08-16 21:02:37	Xandr	L1	Data Scientist	120000	Portland, OR	110000.0	
61803	2021-08-16 22:19:48	Facebook	L4	Data Scientist	233000	Menlo Park, CA	157000.0	

872 rows × 14 columns

In [35]:
```
levels_salary_DS.drop(columns= ['timestamp','level'], inplace=True)
```

```
/var/folders/yn/lfh7s3f52q18zdwkxgbxg58r0000gn/T/ipykernel_7294/327619170.py:1: SettingWithCop
yWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/
indexing.html#returning-a-view-versus-a-copy
  levels_salary_DS.drop(columns= ['timestamp','level'], inplace=True)
```

In [36]:
```
# Fill NaN values with an empty string or any default value
levels_salary_DS['state'].fillna('', inplace=True)
levels_salary_DS['country'].fillna('', inplace=True)


levels_salary_DS
```

```
/var/folders/yn/lfh7s3f52q18zdwkxgbxg58r0000gn/T/ipykernel_7294/4220549634.py:2: SettingWithCo
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/
indexing.html#returning-a-view-versus-a-copy
  levels_salary_DS['state'].fillna('', inplace=True)
/var/folders/yn/lfh7s3f52q18zdwkxgbxg58r0000gn/T/ipykernel_7294/4220549634.py:3: SettingWithCo
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/
indexing.html#returning-a-view-versus-a-copy
  levels_salary_DS['country'].fillna('', inplace=True)
```

Out[36]:

	company	title	totalyearlycompensation	location	basesalary	stockgrantvalue	bonus	ci
23679	Google	Data Scientist	170000	San Francisco, CA	170000.0	0.0	0.0	7
23685	Facebook	Data Scientist	205000	Menlo Park, CA	150000.0	40000.0	15000.0	7
23699	Microsoft	Data Scientist	220000	Bellevue, WA	150000.0	60000.0	10000.0	11
23702	PayPal	Data Scientist	216000	San Jose, CA	160000.0	40000.0	16000.0	7
23724	Amazon	Data Scientist	185000	Cambridge, MA	185000.0	5000.0	0.0	8
...	...	...	...	...	...	...	...	
61592	Netflix	Data Scientist	605000	Los Gatos, CA	605000.0	0.0	0.0	7
61642	Facebook	Data Scientist	185000	Tel Aviv, TA, Israel	133000.0	34000.0	18000.0	15
61687	Adobe	Data Scientist	250000	San Jose, CA	150000.0	100000.0	0.0	7
61793	Xandr	Data Scientist	120000	Portland, OR	110000.0	0.0	10000.0	10
61803	Facebook	Data Scientist	233000	Menlo Park, CA	157000.0	60000.0	16000.0	7

872 rows × 12 columns

In [37]:
```
### I will need to change the country to US where state column has american state abbrevation
### first I found this online where I can create list of us_states
us_states = ["AL", "AK", "AZ", "AR", "CA", "CO", "CT", "DE", "FL", "GA", "HI",
             "ID", "IL", "IN", "IA", "KS", "KY", "LA", "ME", "MD", "MA", "MI",
             "MN", "MS", "MO", "MT", "NE", "NV", "NH", "NJ", "NM", "NY", "NC",
             "ND", "OH", "OK", "OR", "PA", "RI", "SC", "SD", "TN", "TX", "UT",
             "VT", "VA", "WA", "WV", "WI", "WY"]
```

In [38]:
```
### Now I can update the country column to 'US' for rows where the state is a U.S. state abbr
###

levels_salary_DS.loc[levels_salary_DS['state'].isin(us_states), 'country'] = 'United States o


levels_salary_DS
```

Out[38]:

	company	title	totalyearlycompensation	location	basesalary	stockgrantvalue	bonus	ci
23679	Google	Data Scientist	170000	San Francisco, CA	170000.0	0.0	0.0	7
23685	Facebook	Data Scientist	205000	Menlo Park, CA	150000.0	40000.0	15000.0	7
23699	Microsoft	Data Scientist	220000	Bellevue, WA	150000.0	60000.0	10000.0	11
23702	PayPal	Data Scientist	216000	San Jose, CA	160000.0	40000.0	16000.0	7
23724	Amazon	Data Scientist	185000	Cambridge, MA	185000.0	5000.0	0.0	8
...	...	...	...	...	...	...	...	
61592	Netflix	Data Scientist	605000	Los Gatos, CA	605000.0	0.0	0.0	7
61642	Facebook	Data Scientist	185000	Tel Aviv, TA, Israel	133000.0	34000.0	18000.0	15

	company	title	totalyearlycompensation	location	basesalary	stockgrantvalue	bonus	ci
61687	Adobe	Data Scientist	250000	San Jose, CA	150000.0	100000.0	0.0	7
61793	Xandr	Data Scientist	120000	Portland, OR	110000.0	0.0	10000.0	10
61803	Facebook	Data Scientist	233000	Menlo Park, CA	157000.0	60000.0	16000.0	7

872 rows × 12 columns

In [42]:
```
levels_salary_DS['country'].unique()
```

Out[42]:
```
array(['United States of America', 'United Kingdom', 'Singapore',
       'Russia', 'India', '', 'Germany', 'Israel', 'Switzerland',
       'Canada', 'China', 'Ireland', 'Netherlands', 'Ukraine',
       'Australia', 'Luxembourg', 'Poland', 'Sweden', 'Taiwan', 'Japan'],
      dtype=object)
```

In [44]:
```
### I would convert the country names to country abbreviations

### Dictionary mapping country names to alpha-2 codes
country_alpha_2_map = {
    'United States of America': 'US',
    'United Kingdom': 'GB',
    'Singapore': 'SG',
```

```
    'Russia': 'RU',
    'India': 'IN',
    'Germany': 'DE',
    'Israel': 'IL',
    'Switzerland': 'CH',
    'Canada': 'CA',
    'China': 'CN',
    'Ireland': 'IE',
    'Netherlands': 'NL',
    'Ukraine': 'UA',
    'Australia': 'AU',
    'Luxembourg': 'LU',
    'Poland': 'PL',
    'Sweden': 'SE',
    'Taiwan': 'TW',
    'Japan': 'JP'
}

### Map country names to alpha-2 codes using the dictionary
levels_salary_DS['country'] = [country_alpha_2_map.get(country, None) for country in levels_s
levels_salary_DS
### I did this conversion just to be able to merge the datasets if it is necessary
```

```
/var/folders/yn/lfh7s3f52q18zdwkxgbxg58r0000gn/T/ipykernel_7294/1397631321.py:27: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/
indexing.html#returning-a-view-versus-a-copy
  levels_salary_DS['country'] = [country_alpha_2_map.get(country, None) for country in levels_
salary_DS['country']]
```

Out[44]:

	company	title	totalyearlycompensation	location	basesalary	stockgrantvalue	bonus	ci
23679	Google	Data Scientist	170000	San Francisco, CA	170000.0	0.0	0.0	7
23685	Facebook	Data Scientist	205000	Menlo Park, CA	150000.0	40000.0	15000.0	7
23699	Microsoft	Data Scientist	220000	Bellevue, WA	150000.0	60000.0	10000.0	11
23702	PayPal	Data Scientist	216000	San Jose, CA	160000.0	40000.0	16000.0	7
23724	Amazon	Data Scientist	185000	Cambridge, MA	185000.0	5000.0	0.0	8
...	...	...	...	...	...	...	...	
61592	Netflix	Data Scientist	605000	Los Gatos, CA	605000.0	0.0	0.0	7
61642	Facebook	Data Scientist	185000	Tel Aviv, TA, Israel	133000.0	34000.0	18000.0	15
61687	Adobe	Data Scientist	250000	San Jose, CA	150000.0	100000.0	0.0	7
61793	Xandr	Data Scientist	120000	Portland, OR	110000.0	0.0	10000.0	10
61803	Facebook	Data Scientist	233000	Menlo Park, CA	157000.0	60000.0	16000.0	7

872 rows × 12 columns

In [46]:
```
### Now we can drop the location column from dataframe

levels_salary_DS.drop(columns ='location', inplace=True)
levels_salary_DS.head(5)
```

```
/var/folders/yn/lfh7s3f52q18zdwkxgbxg58r0000gn/T/ipykernel_7294/2573104746.py:3: SettingWithCo
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/
indexing.html#returning-a-view-versus-a-copy
  levels_salary_DS.drop(columns ='location', inplace=True)
```

Out[46]:

	company	title	totalyearlycompensation	basesalary	stockgrantvalue	bonus	cityid	dmaid
23679	Google	Data Scientist	170000	170000.0	0.0	0.0	7419	807.0
23685	Facebook	Data Scientist	205000	150000.0	40000.0	15000.0	7300	807.0
23699	Microsoft	Data Scientist	220000	150000.0	60000.0	10000.0	11470	819.0
23702	PayPal	Data Scientist	216000	160000.0	40000.0	16000.0	7422	807.0
23724	Amazon	Data Scientist	185000	185000.0	5000.0	0.0	8821	506.0

In [48]:
```
levels_salary_DS['country'].unique()
```

Out[48]:
```
array(['US', 'GB', 'SG', 'RU', 'IN', None, 'DE', 'IL', 'CH', 'CA', 'CN',
       'IE', 'NL', 'UA', 'AU', 'LU', 'PL', 'SE', 'TW', 'JP'], dtype=object)
```

==========

Now, we finished working on the levels_Fyi_salary_df. I did as much as cleaning in the data. The new data name I will use from now on is "levels_salary_DS" This represents the levels and salaries for only Data Scientists.

In [52]:
```
### Now I will save this cleaned data as csv file.

levels_salary_DS.to_csv('clean_levels_salary_DS.csv', index = False)
```

In [ ]:

In [ ]:

In [ ]: