```
## Importing Libraires
import pandas as pd
import numpy as np
```

```
path='/content/drive/MyDrive/DataSet/Ask A Manager Salary Survey 2019 (Responses).xlsx'
df=pd.read_excel(path,skiprows=1)
```

```
df.head()
```

	Timestamp	How old are you?	What industry do you work in?	Job title	What is your annual salary?	Please indicate the currency	Where are you located? (City/state/country)	How many years of post-college professional work experience do you have?	If your job title needs additional context, please clarify here:	If "Other," please indicate the currency here:
0	2019-04-24 11:43:21.478	35-44	Government	Talent Management Asst. Director	75000	USD	Nashville, TN	11 - 20 years	NaN	NaN
1	2019-04-24 11:43:26.128	25-34	Environmental nonprofit	Operations Director	65000	USD	Madison, Wi	8 - 10 years	NaN	NaN
2	2019-04-24 11:43:26.992	18-24	Market Research	Market Research Assistant	36330	USD	Las Vegas, NV	2 - 4 years	NaN	NaN
3	2019-04-24 11:43:27.379	25-34	Biotechnology	Senior Scientist	34600	GBP	Cardiff, UK	5-7 years	NaN	NaN

Next steps: Generate code with df | New interactive sheet

Basic Information To Check The Nature And Way Of Data

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 35481 entries, 0 to 35480
Data columns (total 10 columns):
 #   Column                                                        Non-Null Count  Dtype
---  ------                                                        --------------  -----
 0   Timestamp                                                     35481 non-null  datetime64[ns]
 1   How old are you?                                              35481 non-null  object
 2   What industry do you work in?                                 34464 non-null  object
 3   Job title                                                     35478 non-null  object
 4   What is your annual salary?                                   35478 non-null  object
 5   Please indicate the currency                                  35481 non-null  object
 6   Where are you located? (City/state/country)                   33633 non-null  object
 7   How many years of post-college professional work experience do you have?  35481 non-null  object
 8   If your job title needs additional context, please clarify here:  7899 non-null  object
 9   If "Other," please indicate the currency here:                307 non-null    object
dtypes: datetime64[ns](1), object(9)
memory usage: 2.7+ MB
```

```
df.shape
```

```
(35481, 10)
```

```
df.columns
```

```
Index(['Timestamp', 'How old are you?', 'What industry do you work in?',
       'Job title', 'What is your annual salary?',
       'Please indicate the currency',
       'Where are you located? (City/state/country)',
       'How many years of post-college professional work experience do you have?',
       'If your job title needs additional context, please clarify here:',
       'If "Other," please indicate the currency here: '],
      dtype='object')
```

```
df.describe()
```

	Timestamp
count	35481
mean	2019-06-24 00:35:59.998689024
min	2019-04-24 11:43:21.478000
25%	2019-04-25 00:09:05.081999872
50%	2019-04-26 23:17:52.987000064
75%	2019-05-05 01:09:20.934000128
max	2025-07-08 13:25:40.392000

```
df.isnull().sum()
```

	0
Timestamp	0
How old are you?	0
What industry do you work in?	1017
Job title	3
What is your annual salary?	3
Please indicate the currency	0
Where are you located? (City/state/country)	1848
How many years of post-college professional work experience do you have?	0
If your job title needs additional context, please clarify here:	27582
If "Other," please indicate the currency here:	35174

dtype: int64

```
df['Timestamp'].unique()
```

```
<DatetimeArray>
['2019-04-24 11:43:21.478000', '2019-04-24 11:43:26.128000',
 '2019-04-24 11:43:26.992000', '2019-04-24 11:43:27.379000',
 '2019-04-24 11:43:28.893000', '2019-04-24 11:43:29.151000',
 '2019-04-24 11:43:29.633000', '2019-04-24 11:43:30.352000',
 '2019-04-24 11:43:34.130000', '2019-04-24 11:43:34.598000',
 ...
 '2024-12-02 12:45:53.412000', '2024-12-02 16:59:18.959000',
 '2024-12-03 10:01:39.176000', '2024-12-11 10:50:53.297000',
 '2024-12-26 16:00:49.764000', '2025-01-02 06:00:18.036000',
 '2025-03-13 13:28:28.189000', '2025-04-28 13:33:51.162000',
 '2025-07-07 11:54:52.351000', '2025-07-08 13:25:40.392000']
Length: 35473, dtype: datetime64[ns]
```

```
df['Job title'].unique()
```

```
array(['Talent Management Asst. Director', 'Operations Director',
       'Market Research Assistant', ..., 'Learning Experience Designer',
       'Senior Frontend Developer', 'Data and Evaluation Manager '],
      dtype=object)
```

```
df['Please indicate the currency'].unique()
```

```
array(['USD', 'GBP', 'CAD', 'EUR', 'SEK', 'Other', 'AUD/NZD', 'JPY',
       'CHF', 'HKD', 'ZAR'], dtype=object)
```

```
df.columns
```

```
Index(['Timestamp', 'How old are you?', 'What industry do you work in?',
       'Job title', 'What is your annual salary?',
       'Please indicate the currency',
       'Where are you located? (City/state/country)',
       'How many years of post-college professional work experience do you have?',
       'If your job title needs additional context, please clarify here:',
```

```
       'If "Other," please indicate the currency here: '],
      dtype='object')
```

First We Rename The Colmns

```
df = df.rename(columns={
    "Timestamp": "timestamp",
    "How old are you?": "age",
    "What industry do you work in?": "industry",
    "Job title": "job_title",
    "Please indicate the currency": "currency",
    "Where are you located? (City/state/country)": "location",
    "How many years of post-college professional work experience do you have?": "experience_years",
    "If your job title needs additional context, please clarify here": "job_title_context",
    'If "Other," please indicate the currency here': "other_currency",
    "What is your annual salary?": "annual_salary"
})

print(df.columns)
```

```
Index(['timestamp', 'age', 'industry', 'job_title', 'annual_salary',
       'currency', 'location', 'experience_years',
       'If your job title needs additional context, please clarify here:',
       'If "Other," please indicate the currency here: '],
      dtype='object')
```

```
df.head()
```

	timestamp	age	industry	job_title	annual_salary	currency	location	experience_years	If your job title needs additional context, please clarify here:	If "Other," please indicate the currency here:
0	2019-04-24 11:43:21.478	35-44	Government	Talent Management Asst. Director	75000	USD	Nashville, TN	11 - 20 years	NaN	NaN
1	2019-04-24 11:43:26.128	25-34	Environmental nonprofit	Operations Director	65000	USD	Madison, Wi	8 - 10 years	NaN	NaN
2	2019-04-24 11:43:26.992	18-24	Market Research	Market Research Assistant	36330	USD	Las Vegas, NV	2 - 4 years	NaN	NaN
3	2019-04-24	25-	Biotechnology	Senior	34600	GBP	Cardiff,	5-7 years	NaN	NaN

Next steps: Generate code with df | New interactive sheet

```
df=df.rename(columns={'If your job title needs additional context, please clarify here:':'job_context',
                'If "Other," please indicate the currency here': 'other_currency'})
```

```
df.head()
```

	timestamp	age	industry	job_title	annual_salary	currency	location	experience_years	job_context	If "Other," please indicate the currency here:
0	2019-04-24 11:43:21.478	35-44	Government	Talent Management Asst. Director	75000	USD	Nashville, TN	11 - 20 years	NaN	NaN
1	2019-04-24 11:43:26.128	25-34	Environmental nonprofit	Operations Director	65000	USD	Madison, Wi	8 - 10 years	NaN	NaN
2	2019-04-24 11:43:26.992	18-24	Market Research	Market Research Assistant	36330	USD	Las Vegas, NV	2 - 4 years	NaN	NaN

Next steps: Generate code with df | New interactive sheet

```
df['age'].unique()
```

```
array(['35-44', '25-34', '18-24', '45-54', '55-64', '65 or over',
       'under 18'], dtype=object)
```

```
df['timestamp'].unique()
```

```
<DatetimeArray>
['2019-04-24 11:43:21.478000', '2019-04-24 11:43:26.128000',
 '2019-04-24 11:43:26.992000', '2019-04-24 11:43:27.379000',
 '2019-04-24 11:43:28.893000', '2019-04-24 11:43:29.151000',
 '2019-04-24 11:43:29.633000', '2019-04-24 11:43:30.352000',
 '2019-04-24 11:43:34.130000', '2019-04-24 11:43:34.598000',
 ...
 '2024-12-02 12:45:53.412000', '2024-12-02 16:59:18.959000',
 '2024-12-03 10:01:39.176000', '2024-12-11 10:50:53.297000',
 '2024-12-26 16:00:49.764000', '2025-01-02 06:00:18.036000',
 '2025-03-13 13:28:28.189000', '2025-04-28 13:33:51.162000',
 '2025-07-07 11:54:52.351000', '2025-07-08 13:25:40.392000']
Length: 35473, dtype: datetime64[ns]
```

```
## Modify Timestamp column
df['timestamp']=pd.to_datetime(df['timestamp'],errors="coerce")
df['year']=df['timestamp'].dt.year
print(df['year'])
```

```
0        2019
1        2019
2        2019
3        2019
4        2019
         ...
35476    2025
35477    2025
35478    2025
35479    2025
35480    2025
Name: year, Length: 35481, dtype: int32
```

```
df.head()
```

	timestamp	age	industry	job_title	annual_salary	currency	location	experience_years	job_context	If "Other," please indicate the currency here:	year
0	2019-04-24 11:43:21.478	35-44	Government	Talent Management Asst. Director	75000	USD	Nashville, TN	11 - 20 years	NaN	NaN	2019
1	2019-04-24 11:43:26.128	25-34	Environmental nonprofit	Operations Director	65000	USD	Madison, Wi	8 - 10 years	NaN	NaN	2019
2	2019-04-24 11:43:26.992	18-24	Market Research	Market Research Assistant	36330	USD	Las Vegas, NV	2 - 4 years	NaN	NaN	2019
3	2019-04-24	25-	Biotechnology	Senior	34600	GBP	Cardiff,	5-7 years	NaN	NaN	2019

Next steps: Generate code with df | New interactive sheet

```
df=df.drop(columns=['timestamp'])
```

```
df['year'].unique()
```

```
array([2019, 2020, 2021, 2022, 2023, 2024, 2025], dtype=int32)
```

```
df['job_context'].unique()
```

```
array([nan,
       'I manage our fundraising department, primarily overseeing our direct mail, planned giving, and grant writing programs.
',
       'equivalent to Assistant Registrar', ...,
       'Environmental, Health and Safety',
       'Managing data team, managing database, using SQL, grant reporting ',
       'Fullstsack expert security specialist'], dtype=object)
```

Handling Missing Values

```
## Checking the missing values
df.isnull().mean()*100
```

	0
age	0.000000
industry	2.866323
job_title	0.008455
annual_salary	0.008455
currency	0.000000
location	5.208421
experience_years	0.000000
job_context	77.737381
If "Other," please indicate the currency here:	99.134748
year	0.000000

dtype: float64

```
##Droping rows in job_title and anual salary
df=df.dropna(subset=['job_title','annual_salary'])
```

```
## We also have missing values in industry column so we fill with  unkown to prevent huge data loss
df['industry']=df['industry'].fillna('unkown')
df['location']=df['location'].fillna('unkown')
```

```
for col in df.columns:
    print(repr(col))
```

```
'age'
'industry'
'job_title'
'annual_salary'
'currency'
'location'
'experience_years'
'job_context'
'If "Other," please indicate the currency here: '
'year'
```

```
## Now we have two columns whhich have missing values greater then 70% we remove those columns directly because they do not hel
df=df.drop(columns=['If "Other," please indicate the currency here: '])
```

```
df['age'].value_counts()
```

age	count
25-34	15829
35-44	10685
45-54	4702
18-24	1968
55-64	1920
65 or over	311
under 18	60

dtype: int64

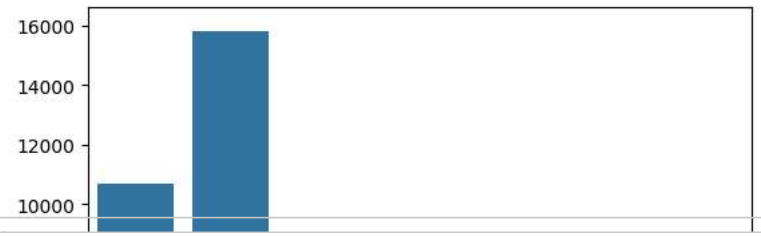
```
df.head()
```

	age	industry	job_title	annual_salary	currency	location	experience_years	job_context	year
0	35-44	Government	Talent Management Asst. Director	75000	USD	Nashville, TN	11 - 20 years	NaN	2019
1	25-34	Environmental nonprofit	Operations Director	65000	USD	Madison, Wi	8 - 10 years	NaN	2019
2	18-24	Market Research	Market Research Assistant	36330	USD	Las Vegas, NV	2 - 4 years	NaN	2019
3	25-34	Biotechnology	Senior Scientist	34600	GBP	Cardiff, UK	5-7 years	NaN	2019
4	25-34	Healthcare	Social worker (embedded in primary	55000	USD	Southeast Michigan, USA	5-7 years	NaN	2019

Next steps: Generate code with df | New interactive sheet

```
import seaborn as sns
sns.countplot(data=df,x='age')
```

```
<Axes: xlabel='age', ylabel='count'>
```




```
from pickle import HIGHEST_PROTOCOL
## Transforming Age Column with midpoint
## Creating the function which takes the value
import numpy as np

def get_midpoint( age_str):
  if "-" in age_str:
    low, high= age_str.split("-")
    return (int(low)+int(high))/2
  elif "under" in age_str.lower():
    return 17

  elif "over" in age_str.lower():
    return 70

  else:
     return np.nan

df['age_num']=df['age'].apply(get_midpoint)
  ## Now we check the result
print(df[['age','age_num']].head())
```

```
     age  age_num
0  35-44     39.5
1  25-34     29.5
2  18-24     21.0
3  25-34     29.5
4  25-34     29.5
```

```
df['age_num'].unique()
```

```
array([39.5, 29.5, 21. , 49.5, 59.5, 70. , 17. ])
```

```
df.drop(columns=['age'],inplace=True)
```

```
df.head()
```

	industry	job_title	annual_salary	currency	location	experience_years	job_context	year	age_num
0	Government	Talent Management Asst. Director	75000	USD	Nashville, TN	11 - 20 years	NaN	2019	39.5
1	Environmental nonprofit	Operations Director	65000	USD	Madison, Wi	8 - 10 years	NaN	2019	29.5
2	Market Research	Market Research Assistant	36330	USD	Las Vegas, NV	2 - 4 years	NaN	2019	21.0
3	Biotechnology	Senior Scientist	34600	GBP	Cardiff, UK	5-7 years	NaN	2019	29.5

Next steps: Generate code with df | New interactive sheet

Transforming Industry Column

```
## First we make the copy of our original industry column
df['industry_clean']=df['industry'].copy()
## We create another copy  in which we do changes
df['industry_work']=df['industry_clean'].astype(str)
```

```
df['industry_work'] = df['industry_work'].replace(r'[/|,;\\]', ' ', regex=True)
df['industry_work'] = df['industry_work'].replace(r'[\(\)\[\]]', ' ', regex=True)
# normalize different dash types into '-' (optional)
df['industry_work'] = df['industry_work'].replace(r'[—–]', '-', regex=True)
```

```
##Converting into lower case
df['industry_work']=df['industry_work'].str.lower()
##remove spaces
df['industry_work']=df['industry_work'].str.strip()
df['industry_work'] = df['industry_work'].replace(r'\s+', ' ', regex=True).str.strip()
```

```
## Now we keep &and - and other irrelevant words or characters i remove it
import re
df['industry_work'] = df['industry_work'].apply(lambda x: re.sub(r'[^a-z\s&-]', '', str(x)))
```

```
manual_map = {
  'unkown': 'unknown',
  'manufaturing': 'manufacturing',
  'it': 'technology',
  'tech': 'technology',
  'health care': 'healthcare',
  'education health care': 'education & healthcare',
  'non profit': 'nonprofit',

}
df['industry_work'] = df['industry_work'].replace(manual_map)
```

```
df['industry_work'].value_counts()
```

industry_work	count
technology	2689
healthcare	1570
education	1195
higher education	1035
unknown	1013
...	...
tech--software	1
home improvement retail	1
urban planning consulting	1
grocery foodservice wholesale	1
transportation & travel	1

5988 rows × 1 columns

dtype: int64

```
## As we see there is very high cardinality and  to overcome this we reduce the cardinality
counts= df['industry_work'].value_counts()
keep=counts[counts>=50].index
df['industry_reduced']=df['industry_work'].where(df['industry_work'].isin(keep),other='other')
```

```
print("Unique Clean:", df['industry_clean'].nunique())
print("Unique work:", df['industry_work'].nunique())
print("Unique reduced:", df['industry_reduced'].nunique())
print(df[['industry_clean','industry_work','industry_reduced']].sample(20))
```

```
Unique Clean: 7932
Unique work: 5988
Unique reduced: 91
                      industry_clean                   industry_work  \
4912                 Higher Education                higher education
```

```
31874                          Food                          food
13173                     Libraries                     libraries
24124                    Healthcare                    healthcare
28987                    Technology                    technology
1332                Human Resources                human resources
5970   Higher Education Administration  higher education administration
6587                     Education                     education
15228              Public Accounting              public accounting
4435                     Architecture                   architecture
32935                       Pharmacy                       pharmacy
30616                    Healthcare                    healthcare
31557                    Healthcare                    healthcare
284                       Marketing                     marketing
29841                    Education                     education
3982                Higher Education                higher education
27618                    Healthcare                    healthcare
647                         BioTech                       biotech
29615                            IT                    technology
25497                        Mining                        mining

       industry_reduced
4912    higher education
31874             other
13173         libraries
24124        healthcare
28987        technology
1332     human resources
5970              other
6587          education
15228  public accounting
4435        architecture
32935             other
30616        healthcare
31557        healthcare
284           marketing
29841         education
3982    higher education
27618        healthcare
647             biotech
29615        technology
25497             other
```

```
## Now We keep only one column which is reduced
df=df.drop(columns='industry')
```

```
df=df.rename(columns={'industry_reduced':'industry'})
```

```
## Now we remove other two remainng colmns clean and work
df=df.drop(columns=['industry_clean','industry_work'])
```

```
df['industry'].unique()
```

```
array(['government', 'other', 'market research', 'biotechnology',
       'healthcare', 'nonprofit', 'higher education', 'libraries',
       'mental health', 'telecommunications', 'education', 'higher ed',
       'communications', 'media', 'publishing', 'non-profit',
       'technology', 'construction', 'academia', 'consulting', 'banking',
       'journalism', 'health insurance', 'public libraries',
       'financial services', 'marketing', 'finance', 'accounting', 'law',
       'unknown', 'legal', 'manufacturing', 'insurance', 'library',
       'oil and gas', 'hospitality', 'biotech', 'civil engineering',
       'public library', 'advertising', 'retail', 'public accounting',
       'architecture', 'software', 'research', 'telecom', 'health',
       'fintech', 'medicine', 'digital marketing', 'automotive',
       'social services', 'transportation', 'environmental consulting',
       'utilities', 'defense', 'medical device', 'pharmaceuticals',
       'government contracting', 'engineering', 'real estate',
       'aerospace', 'public relations', 'pharma',
       'information technology', 'federal government', 'public health',
       'financial', 'oil & gas', 'design', 'medical devices',
       'local government', 'energy', 'state government',
       'human resources', 'healthcare it', 'medical',
       'management consulting', 'museums', 'software development',
       'professional services', 'law enforcement', 'entertainment',
       'sales', 'pharmaceutical', 'philanthropy', 'aviation',
       'semiconductor', 'restaurant', 'agriculture', 'logistics'],
      dtype=object)
```

```
df['industry'].value_counts().head(30)
```

industry	count
other	12332
technology	2689
healthcare	1570
education	1195
higher education	1035
unknown	1013
nonprofit	880
software	804
government	771
finance	720
manufacturing	622
legal	532
insurance	510
retail	473
construction	468
law	412
academia	370
non-profit	370
consulting	361
marketing	360
banking	337
financial services	318
publishing	296
information technology	280
media	268
advertising	252
engineering	236
hospitality	229
automotive	223
real estate	221

dtype: int64

```
df['industry'].unique()
```

```
array(['government', 'other', 'market research', 'biotechnology',
       'healthcare', 'nonprofit', 'higher education', 'libraries',
       'mental health', 'telecommunications', 'education', 'higher ed',
       'communications', 'media', 'publishing', 'non-profit',
       'technology', 'construction', 'academia', 'consulting', 'banking',
       'journalism', 'health insurance', 'public libraries',
       'financial services', 'marketing', 'finance', 'accounting', 'law',
       'unknown', 'legal', 'manufacturing', 'insurance', 'library',
       'oil and gas', 'hospitality', 'biotech', 'civil engineering',
       'public library', 'advertising', 'retail', 'public accounting',
       'architecture', 'software', 'research', 'telecom', 'health',
       'fintech', 'medicine', 'digital marketing', 'automotive',
       'social services', 'transportation', 'environmental consulting',
       'utilities', 'defense', 'medical device', 'pharmaceuticals',
       'government contracting', 'engineering', 'real estate',
       'aerospace', 'public relations', 'pharma',
       'information technology', 'federal government', 'public health',
```

```
       'financial', 'oil & gas', 'design', 'medical devices',
       'local government', 'energy', 'state government',
       'human resources', 'healthcare it', 'medical',
       'management consulting', 'museums', 'software development',
       'professional services', 'law enforcement', 'entertainment',
       'sales', 'pharmaceutical', 'philanthropy', 'aviation',
       'semiconductor', 'restaurant', 'agriculture', 'logistics'],
      dtype=object)
```

```
##Now we have some mapping need so we mapp some values
industry_mapping={
    'higher education': 'education',
    'academia': 'education',
      'non-profit': 'nonprofit',
     'law': 'legal',
        'banking': 'finance',
    'financial services': 'finance',
     'unkown': 'unknown'

}
df['industry']=df['industry'].replace(industry_mapping)
```

```
# keep only top 20 industries, rest as "other"
top_industries = df['industry'].value_counts().nlargest(20).index
df['industry'] = df['industry'].apply(lambda x: x if x in top_industries else 'other')
```

```
df['industry'].unique()
```

```
array(['government', 'other', 'healthcare', 'nonprofit', 'education',
       'media', 'publishing', 'technology', 'construction', 'consulting',
       'finance', 'marketing', 'legal', 'unknown', 'manufacturing',
       'insurance', 'advertising', 'retail', 'software',
       'information technology'], dtype=object)
```

```
df['industry'].value_counts()
```

industry	count
other	18569
technology	2689
education	2600
healthcare	1570
finance	1375
nonprofit	1250
unknown	1013
legal	944
software	804
government	771
manufacturing	622
insurance	510
retail	473
construction	468
consulting	361
marketing	360
publishing	296
information technology	280
media	268
advertising	252

dtype: int64

```
df.head()
```

	job_title	annual_salary	currency	location	experience_years	job_context	year	age_num	industry
0	Talent Management Asst. Director	75000	USD	Nashville, TN	11 - 20 years	NaN	2019	39.5	government
1	Operations Director	65000	USD	Madison, Wi	8 - 10 years	NaN	2019	29.5	other
2	Market Research Assistant	36330	USD	Las Vegas, NV	2 - 4 years	NaN	2019	21.0	other
3	Senior Scientist	34600	GBP	Cardiff, UK	5-7 years	NaN	2019	29.5	other

Next steps: Generate code with df | New interactive sheet

Transforming Job Title Column

```
## First we get the basic overview and info of job title column
print(df['job_title'].value_counts())
```

```
job_title
Project Manager                                                385
Director                                                       369
Software Engineer                                              333
Manager                                                        313
Teacher                                                        200
                                                              ...
Clinic coordinator                                               1
Communications coordinator/Sustainability Information Curator    1
Executive Assistant (to Chief Business Officer)                  1
Patient Service Specialist                                       1
Senior Process Engineer                                          1
Name: count, Length: 16272, dtype: int64
```

```
df['job_title'].head(40)
```

	job_title
0	Talent Management Asst. Director
1	Operations Director
2	Market Research Assistant
3	Senior Scientist
4	Social worker (embedded in primary care)
5	Associate Consultant
6	Development Manager
7	Student Records Coordinator
8	Director
9	Copywriter
10	office manager
11	HR and Administrative Coordinator
12	Marketing Manager
13	Senior research executive
14	Teacher
15	Senior Consultant
16	Associate Professor
17	communiity investment officer
18	Visual Effects Compositor
19	senior marketing manager
20	Assistant Media Editor
21	Assistant Director of Development Research
22	Public Services Coordinator
23	Instructional Consultant
24	Event Producer
25	Support QA Lead
26	Associate Governmental Program Analyst
27	convention coordinator
28	Pet sitter
29	Senior Research Scientist
30	Department Head
31	Grant Writer & Planner
32	Catering Lead
33	Human Resources Manager
34	Property Manager
35	Business Analyst
36	Senior Editor
37	Content Marketing Specialist
38	Personnel Coordinator
39	Content Editor

dtype: object

```
df['job_title'].value_counts().head(20)
```

job_title	count
Project Manager	385
Director	369
Software Engineer	333
Manager	313
Teacher	200
Attorney	191
Program Manager	189
Analyst	184
Librarian	138
Associate	135
Engineer	132
Product Manager	132
Operations Manager	129
Consultant	128
Marketing Manager	125
Senior Software Engineer	118
Software Developer	117
Administrative Assistant	117
Executive Assistant	112
Office Manager	110

dtype: int64

```
df['job_title'].duplicated().sum()
```

```
np.int64(19203)
```

```
df.duplicated().sum()
```

```
np.int64(923)
```

```
## We remove duplictates whole dataset
df=df.drop_duplicates()
```

```
df.duplicated().sum()
```

```
np.int64(0)
```

Steps To Clean the job title column

```
## First We Convert into lower case
df['job_title_clean']=df['job_title'].str.lower()
```

```
df['job_title_clean'].value_counts()
```

job_title_clean	count
project manager	458
software engineer	425
director	373
manager	328
teacher	215
...	...
business development administrator	1
principal technical writer	1
intermediate rater	1
research & evaluation lead	1
lead community & social media manager	1

14492 rows × 1 columns

dtype: int64

```
## Now we remove the job title column
df=df.drop(columns='job_title')
```

```
## Removing Extra leading spaces and special characters
df['job_title_clean']=df['job_title_clean'].str.strip()
df['job_title_clean'] = df['job_title_clean'].str.replace(r'[/\-,().]', ' ', regex=True)
df['job_title_clean'] = df['job_title_clean'].str.replace(r'\s+', ' ', regex=True).str.strip()
```

```
df['job_title_clean'].value_counts()
```

job_title_clean	count
project manager	525
software engineer	461
director	446
manager	367
attorney	237
...	...
program leader	1
love spell: to return the husband to enchant the wife the lapel of a rival various conspiracies cleaning energy from the evil eye and spoilage psychic esoteric tarot reading an actor any job any experiments it computer technology website creation	1
charitable giving coordinator	1
cumbumdaniel	1
family specialist	1

13263 rows × 1 columns

dtype: int64

```
df['job_title_clean'].nunique()
```

```
13263
```

```
df['job_title_clean'].unique()
```

```
array(['talent management asst director', 'operations director',
       'market research assistant', ..., 'learning experience designer',
       'senior frontend developer', 'data and evaluation manager'],
      dtype=object)
```

```
# Find job titles that appear less than 5 times
rare_titles = df['job_title_clean'].value_counts()[df['job_title_clean'].value_counts() < 10].index

# Replace them with "other"
df['job_title_clean'] = df['job_title_clean'].replace(rare_titles, 'other')
```

```
df['job_title_clean'].value_counts()
```

job_title_clean	count
other	18464
project manager	525
software engineer	461
director	446
manager	367
...	...
psychologist	10
cashier	10
trader	10
chief engineer	10
md	10

440 rows × 1 columns

dtype: int64

```
###Now we rename the column
df.rename(columns={'job_title_clean':"job_title"},inplace=True)
```

```
df.head()
```

	annual_salary	currency	location	experience_years	job_context	year	age_num	industry	job_title
0	75000	USD	Nashville, TN	11 - 20 years	NaN	2019	39.5	government	other
1	65000	USD	Madison, Wi	8 - 10 years	NaN	2019	29.5	other	operations director
2	36330	USD	Las Vegas, NV	2 - 4 years	NaN	2019	21.0	other	other
3	34600	GBP	Cardiff, UK	5-7 years	NaN	2019	29.5	other	senior scientist

Next steps: Generate code with df | New interactive sheet

```
df['job_title'].unique()
```

```
       'photographer', 'statistician', 'safety manager',
       'senior developer', 'production manager', 'relationship manager',
       'program analyst', 'account supervisor', 'training manager',
       'mechanical engineer', 'advisor', 'product analyst', 'writer',
       'rn', 'controls engineer', 'art director',
       'staff software engineer', 'development coordinator',
       'nurse practitioner', 'phd student', 'branch manager',
       'procurement manager', 'assistant controller',
       'senior financial analyst', 'development officer', 'audit manager',
       'financial advisor', 'intern', 'senior vice president',
       'sr financial analyst', 'assistant store manager',
       'senior director', 'analytics manager', 'qa analyst',
       'associate product manager', 'chief financial officer',
       'site manager', 'engagement manager', 'postdoctoral fellow', 'ceo',
       'marketing analyst', 'director of product management',
       'it project manager', 'electrical engineer', 'professor',
       'auditor', 'sales engineer', 'cto', 'administrative officer',
       'technical director', 'loan officer', 'principal consultant',
       'police officer', 'physician assistant', 'director of engineering',
       'software development manager', 'optometrist',
       'school psychologist', 'legal counsel', 'assistant vice president',
       'sr engineer', 'server', 'training specialist',
       'occupational therapist', 'operations analyst',
       'system administrator', 'driver', 'sales director',
       'staff engineer', 'marketing', 'it specialist',
       'logistics manager', 'sales representative',
       'principal software engineer', 'asset manager', 'sales executive',
       'doctor', 'front end developer', 'psychologist',
       'industrial engineer', 'electrician', 'sales', 'cashier',
       'security analyst', 'information security analyst',
       'solution architect', 'chief operating officer',
       'sr product manager', 'devops engineer', 'nurse',
       'network administrator', 'system engineer', 'svp', 'pilot',
       'sr software engineer', 'interior designer',
       'regional sales manager', 'cio', 'director of it', 'dentist',
       'technology manager', 'trader', 'sr program manager', nan,
       'purchasing manager', 'chief engineer', 'superintendent', 'md'],
      dtype=object)
```

Now we transform the Currency column

```
df['currency'].unique()
```

```
array(['USD', 'GBP', 'CAD', 'EUR', 'SEK', 'Other', 'AUD/NZD', 'JPY',
       'CHF', 'HKD', 'ZAR'], dtype=object)
```

```
df['currency'].value_counts()
```

currency	count
USD	30284
CAD	1733
GBP	1035
AUD/NZD	643
EUR	524
Other	201
CHF	39
JPY	35
SEK	27
ZAR	16
HKD	15

dtype: int64

```
df.head()
```

	annual_salary	currency	location	experience_years	job_context	year	age_num	industry	job_title
0	75000	USD	Nashville, TN	11 - 20 years	NaN	2019	39.5	government	other
1	65000	USD	Madison, Wi	8 - 10 years	NaN	2019	29.5	other	operations director
2	36330	USD	Las Vegas, NV	2 - 4 years	NaN	2019	21.0	other	other
3	34600	GBP	Cardiff, UK	5-7 years	NaN	2019	29.5	other	senior scientist

Next steps: Generate code with df | New interactive sheet

we make the copy of origial column

```
df['currency_raw']=df['currency'].copy()
```

```
## We transform the currency column  convert into upper case

df['currency']=df['currency'].str.strip().str.upper()
```

```
## Now we replace the column name AUD/NZD with AUD
df['currency']=df['currency'].replace('AUD/NZD','AUD')
```

```
df['currency'].value_counts()
```

currency	count
USD	30284
CAD	1733
GBP	1035
AUD	643
EUR	524
OTHER	201
CHF	39
JPY	35
SEK	27
ZAR	16
HKD	15

dtype: int64

```
## There are rare currencies now we decide threshhold having less then 50 currencies into other
rare_currencies=df['currency'].value_counts()[df['currency'].value_counts()<50].index
df['currency']=df['currency'].replace(rare_currencies,'OTHER')
```

```
df['currency'].value_counts()
```

currency	count
USD	30284
CAD	1733
GBP	1035
AUD	643
EUR	524
OTHER	333

dtype: int64

```
## Dropping the currency_raw column
df=df.drop(columns='currency_raw')
```

Now we apply modifications on location

```
## Making the location column into  lower case and remove spaces
df['location'] = df['location'].str.lower().str.strip()
df['location'] = df['location'].str.replace(r'\s+', ' ', regex=True)
```

```
## Replace unknown with nan
import numpy as np
df['location'] = df['location'].replace(['unknown','unkown','na','n/a'], np.nan)
```

```
## Splity City ,state, country
df[['city','state','country']] = df['location'].str.split(',', n=2, expand=True)
```

```
df['city'] = df['city'].str.strip()
df['state'] = df['state'].str.strip()
df['country'] = df['country'].str.strip()
```

```
df['country'].unique()
```

```
array([None, 'united states', 'usa', 'canada', nan,
       'united states of america', 'us', 'u.s.', 'somerset', '23222',
       'new zealand', 'usa (sf bay area)', 'uk', 'u.s.a.', 'ca', '07054',
       'australia', 'but work on road',
       'but company is based in sarasota, fl)', 'usa (job is remote)',
       'aus', '19119', 'vanderburgh county',
       'usa (company is located in brooklyn, new york, usa)', 'america',
       'canada me - ottawa, on, canada', 'us.', 'usa (chicago suburb)',
       'can', 'usa (company headquarters in rockville, md, usa)', 'mn',
       'usa (company location. i work remotely from the east coast.)',
       '72205', 'mexico', 'south africa', '02143', 'usa (northern va)',
       '60657', 'germany', 'il', 'tx', 'brevard', 'oh', 'england', 'uss',
       'usa (dc metro area)', 'isa', 'united kingdom',
       'westchester county', 'austrakia', 'usa (remote worker)',
       'corp location is nj', 'wa', 'japan', '20003', 'usa.',
       'usa - but company is remote and international', 'the netherlands',
       'ua', 'usw', 'sweden', '87505', 'arlington county va', 'ny',
       'maricopa', 'hinterm mond', '27701', 'hidalgo',
       'il, and oh plants.', 'united stares', 'philippines',
       'united states of americarica', 'us', 'u.s.a', 'usa metro area',
       'orange county', 'jefferson', 'use', 'maastricht',
       'usa (remote position)', 'uas',
       'usa (remote for boston, ma company)', 'nc',
       'saskatchewan, canada', 'united state', 'ma, usa', 'ysa', 'uda',
       'usa (wash. dc region)', 'europe', '02215', 'usa6', 'utah county',
       '63132', 'romania', 'w.a', '91311', 'canada (us company)', '22902',
       'md usa', 'brazil', '', '22314', 'unites states', 'france',
       'poland', 'india'], dtype=object)
```

```
## First we normalize and make it consistent  country
df['country'].value_counts(dropna=False).head(80)
```

country	count
None	25936
usa	5467
NaN	1803
canada	497
us	329
...	...
usa metro area	1
orange county	1
use	1
jefferson	1
usa (remote position)	1

80 rows × 1 columns

dtype: int64

```
df['country_clean'] = df['country'].astype(str).str.lower().str.strip()
# remove leading/trailing punctuation
df['country_clean'] = df['country_clean'].str.replace(r'^[^\w]+|[^\w]+$', '', regex=True)
```

```
df['country_clean'] = df['country_clean'].str.replace(r'\(.*\)', '', regex=True).str.strip()
```

```
df.loc[df['country_clean'].str.fullmatch(r'\d+'), 'country_clean'] = ''
df['country_clean'] = df['country_clean'].replace('', np.nan)
```

```
mapping = {
  # United States variants
  'usa':'United States','us':'United States','u.s.':'United States','u.s.a.':'United States',
  'united states':'United States','united states of america':'United States','united state':'United States',
  'united stares':'United States','america':'United States','usa.':'United States','us.':'United States',
  'us':'United States','usa (remote)':'United States','usa (sf bay area)':'United States',
  # State tokens that crept in
  'ny':'United States','nyc':'United States','tx':'United States','ca':'United States','wa':'United States','ma':'United States
  # Canada
  'can':'Canada','ca usa':'Canada','canada':'Canada','saskatchewan, canada':'Canada','vancouver canada':'Canada',
  # UK / England
  'uk':'United Kingdom','england':'United Kingdom','united kingdom':'United Kingdom',
  # common countries
  'australia':'Australia','aus':'Australia','new zealand':'New Zealand','nz':'New Zealand',
  'mexico':'Mexico','south africa':'South Africa','germany':'Germany','france':'France','poland':'Poland','india':'India','braz
  # handle silly or long forms
  'canada me - ottawa, on, canada':'Canada'
}
df['country_clean'] = df['country_clean'].replace(mapping).fillna(df['country_clean'])
# standardize capitalization
df['country_clean'] = df['country_clean'].where(df['country_clean'].isna(), df['country_clean'].str.title())
```

```
US_states = {...}  # set of two-letter US state codes, e.g. 'NY','CA','TX',...
mask = df['country_clean'].isna() & df['state'].str.upper().isin(US_states).fillna(False)
df.loc[mask, 'country_clean'] = 'United States'
# also infer Canada from province codes if you track them
```

```
counts = df['country_clean'].value_counts()
rare = counts[counts < 20].index   # threshold: 20 (pick 5/10/20 based on data)
```

```
df['country_ml'] = df['country_clean'].where(~df['country_clean'].isin(rare), 'Other')
```

```
df['country_clean'].value_counts()
df[['country','country_clean','country_ml']].sample(20, random_state=1)
```

	country	country_clean	country_ml
35080	usa	United States	United States
18467	None	None	None
24218	NaN	Nan	Nan
24642	None	None	None
27244	usa	United States	United States
23446	None	None	None
19178	None	None	None
15028	None	None	None
20959	usa	United States	United States
27579	None	None	None
6784	usa	United States	United States
13360	None	None	None
24619	usa	United States	United States
27632	None	None	None
19093	NaN	Nan	Nan
676	None	None	None
34720	None	None	None
7643	None	None	None
28841	None	None	None
1409	None	None	None

```
df['country_ml'].value_counts()
```

country_ml	count
None	25936
United States	6115
Nan	1803
Canada	500
Other	94
Australia	65
United Kingdom	23

dtype: int64

```
df.head()
```

	annual_salary	currency	location	experience_years	job_context	year	age_num	industry	job_title	city	state	countr
0	75000	USD	nashville, tn	11 - 20 years	NaN	2019	39.5	government	other	nashville	tn	Nor
1	65000	USD	madison, wi	8 - 10 years	NaN	2019	29.5	other	operations director	madison	wi	Nor
2	36330	USD	las vegas, nv	2 - 4 years	NaN	2019	21.0	other	other	las vegas	nv	Nor
3	34600	GBP	cardiff, uk	5-7 years	NaN	2019	29.5	other	senior scientist	cardiff	uk	Nor
4	55000	USD	southeast michigan, usa	5-7 years	NaN	2019	29.5	healthcare	other	southeast michigan	usa	Nor

Next steps: Generate code with df New interactive sheet

```
##We remove the location column
df.drop(columns=['location','country'],inplace=True)
```

```
df.head()
```

	annual_salary	currency	experience_years	job_context	year	age_num	industry	job_title	city	state	country_clean	c
0	75000	USD	11 - 20 years	NaN	2019	39.5	government	other	nashville	tn	None	
1	65000	USD	8 - 10 years	NaN	2019	29.5	other	operations director	madison	wi	None	
2	36330	USD	2 - 4 years	NaN	2019	21.0	other	other	las vegas	nv	None	
3	34600	GBP	5-7 years	NaN	2019	29.5	other	senior scientist	cardiff	uk	None	
4	55000	USD	5-7 years	NaN	2019	29.5	healthcare	other	southeast	usa	None	

Next steps: Generate code with df New interactive sheet

```
df.head()
```

	annual_salary	currency	experience_years	job_context	year	age_num	industry	job_title	city	state	country_clean	c
0	75000	USD	11 - 20 years	NaN	2019	39.5	government	other	nashville	tn	None	
1	65000	USD	8 - 10 years	NaN	2019	29.5	other	operations director	madison	wi	None	
2	36330	USD	2 - 4 years	NaN	2019	21.0	other	other	las vegas	nv	None	
3	34600	GBP	5-7 years	NaN	2019	29.5	other	senior scientist	cardiff	uk	None	
4	55000	USD	5-7 years	NaN	2019	29.5	healthcare	other	southeast	usa	None	

Next steps: Generate code with df New interactive sheet

```
df.columns
```

```
Index(['annual_salary', 'currency', 'experience_years', 'job_context', 'year',
       'age_num', 'industry', 'job_title', 'city', 'state', 'country_clean',
       'country_ml'],
      dtype='object')
```

```
df.drop(columns=['country', 'country_clean'], inplace=True, errors='ignore')
```

```
df.head()
```

	annual_salary	currency	experience_years	job_context	year	age_num	industry	job_title	city	state	country_ml
0	75000	USD	11 - 20 years	NaN	2019	39.5	government	other	nashville	tn	None
1	65000	USD	8 - 10 years	NaN	2019	29.5	other	operations director	madison	wi	None
2	36330	USD	2 - 4 years	NaN	2019	21.0	other	other	las vegas	nv	None
3	34600	GBP	5-7 years	NaN	2019	29.5	other	senior scientist	cardiff	uk	None

Next steps: Generate code with df | New interactive sheet

Transforming Experience Years column and gain some insights

```
df['experience_years'].unique()
```

```
array(['11 - 20 years', '8 - 10 years', '2 - 4 years', '5-7 years',
       '21 - 30 years', '1 year or less', '41 years or more',
       '31 - 40 years'], dtype=object)
```

```
df['experience_years'].value_counts()
```

experience_years	count
11 - 20 years	9439
5-7 years	6429
8 - 10 years	6010
2 - 4 years	5512
21 - 30 years	3939
1 year or less	1758
31 - 40 years	1249
41 years or more	216

dtype: int64

```
## Create new column for ML by converting experience to numeric
def convert_experience_to_numeric(value):
    # Handle missing values
    if pd.isna(value):
        return np.nan

    # Clean text (remove spaces, lowercase)
    value = value.lower().strip()

    # Handle special cases
    if "less" in value:
        return 0.5
    elif "more" in value:
        return 45

    # Extract numeric parts
    numbers = [int(num) for num in value.split() if num.isdigit()]

    # Handle ranges or single values
    if len(numbers) == 2:
        return (numbers[0] + numbers[1]) / 2
    elif len(numbers) == 1:
        return numbers[0]
    else:
        return np.nan
```

```
## Know we covert and apply convert it into numeric
df['experience_years_numeric'] = df['experience_years'].apply(convert_experience_to_numeric)
```

```
df.head()
```

	annual_salary	currency	experience_years	job_context	year	age_num	industry	job_title	city	state	country_ml	expe
0	75000	USD	11 - 20 years	NaN	2019	39.5	government	other	nashville	tn	None	
1	65000	USD	8 - 10 years	NaN	2019	29.5	other	operations director	madison	wi	None	
2	36330	USD	2 - 4 years	NaN	2019	21.0	other	other	las vegas	nv	None	
3	34600	GBP	5-7 years	NaN	2019	29.5	other	senior scientist	cardiff	uk	None	
4	55000	USD	5-7 years	NaN	2019	29.5	healthcare	other	southeast michigan	usa	None	

Next steps: Generate code with df | New interactive sheet

Transforming Job Context Column

```
df['job_context'].unique()
```

```
array([nan,
       'I manage our fundraising department, primarily overseeing our direct mail, planned giving, and grant writing programs.
',
       'equivalent to Assistant Registrar', ...,
       'Environmental, Health and Safety',
       'Managing data team, managing database, using SQL, grant reporting ',
       'Fullstsack expert security specialist'], dtype=object)
```

```
df['job_context'].value_counts()
```

job_context	count
Sales	30
Attorney	14
Project Manager	12
IT	12
Fundraising	10
...	...
I work on larger city land use plans and projects that have a longer implementation. I can't help you get the plans for your local business approved.	1
Tech support/customer support	1
Full charge bookkeeper currently doing heavy payroll w/third party vendor	1
district director for a state senator	1
"salary" here includes bonus and stock	1

7298 rows × 1 columns

```
## As job_context has 77% missing values, we drop it
df.drop(columns=['job_context'], inplace=True)
```

```
df.head()
```

	annual_salary	currency	experience_years	year	age_num	industry	job_title	city	state	country_ml	experience_years
0	75000	USD	11 - 20 years	2019	39.5	government	other	nashville	tn	None	
1	65000	USD	8 - 10 years	2019	29.5	other	operations director	madison	wi	None	
2	36330	USD	2 - 4 years	2019	21.0	other	other	las vegas	nv	None	
3	34600	GBP	5-7 years	2019	29.5	other	senior scientist	cardiff	uk	None	
4	55000	USD	5-7 years	2019	29.5	healthcare	other	southeast	usa	None	

Next steps: Generate code with df | New interactive sheet

There is a need some aligning in annual salary column

```
df['annual_salary'].unique()
```

```
array([75000, 65000, 36330, ..., '90,000 + 12% bonus', 266000, '117, 000'],
      dtype=object)
```

```
df['annual_salary'].value_counts()
```

annual_salary	count
60000	803
80000	776
70000	719
65000	701
90000	686
...	...
186070	1
$O62,000	1
73 000$	1
54371	1
36330	1

4285 rows × 1 columns

dtype: int64

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

def clean_salary(value):
    if pd.isna(value):
        return np.nan

    # Convert to string
    value = str(value).lower()

    # Remove currency symbols and commas
    value = re.sub(r'[\$,€£]', '', value)
    value = value.replace(',', '').replace('usd', '').replace('cad', '').replace('aud', '')

    # Replace common letter typos (O → 0)
    value = value.replace('o', '0')

    # Extract first number found in string
    match = re.search(r'\d+', value)
    if match:
        return float(match.group())
    else:
```

```
        return np.nan

# Apply to column
df['annual_salary_ml'] = df['annual_salary'].apply(clean_salary)

# Check few rows
print(df[['annual_salary', 'annual_salary_ml']].head(15))
```

```
   annual_salary  annual_salary_ml
0          75000           75000.0
1          65000           65000.0
2          36330           36330.0
3          34600           34600.0
4          55000           55000.0
5          45000           45000.0
6          51000           51000.0
7          54371           54371.0
8          45000           45000.0
9          80000           80000.0
10         72000           72000.0
11         50000           50000.0
12       73 000$              73.0
13         30000           30000.0
14         54000           54000.0
```

```
df.head()
```

	annual_salary	currency	experience_years	year	age_num	industry	job_title	city	state	country_ml	experience_years
0	75000	USD	11 - 20 years	2019	39.5	government	other	nashville	tn	None	
1	65000	USD	8 - 10 years	2019	29.5	other	operations director	madison	wi	None	
2	36330	USD	2 - 4 years	2019	21.0	other	other	las vegas	nv	None	
3	34600	GBP	5-7 years	2019	29.5	other	senior scientist	cardiff	uk	None	
4	55000	USD	5-7 years	2019	29.5	healthcare	other	southeast michigan	usa	None	

Next steps: Generate code with df | New interactive sheet

```
## Drop annual_salary column
df.drop(columns=['annual_salary'], inplace=True)
```

```
df.head()
```

	currency	experience_years	year	age_num	industry	job_title	city	state	country_ml	experience_years_numeric	annua
0	USD	11 - 20 years	2019	39.5	government	other	nashville	tn	None	15.5	
1	USD	8 - 10 years	2019	29.5	other	operations director	madison	wi	None	9.0	
2	USD	2 - 4 years	2019	21.0	other	other	las vegas	nv	None	3.0	
3	GBP	5-7 years	2019	29.5	other	senior scientist	cardiff	uk	None	NaN	
4	USD	5-7 years	2019	29.5	healthcare	other	southeast michigan	usa	None	NaN	

Next steps: Generate code with df | New interactive sheet

```
df['annual_salary_ml'].sort_values(ascending=False).head(10)
```

	annual_salary_ml
14472	2.000000e+28
34057	3.000000e+08
26325	2.080000e+08
11791	1.320000e+08
35476	1.000000e+08
5865	1.000000e+08
10836	9.000000e+07
13559	8.200000e+07
32343	5.520000e+07
16950	4.800000e+07

dtype: float64

```
upper_limit = df['annual_salary_ml'].quantile(0.99)
df = df[df['annual_salary_ml'] <= upper_limit]
```

```
df['annual_salary_ml'].describe()
```

	annual_salary_ml
count	34197.000000
mean	91010.848437
std	60437.798398
min	0.000000
25%	53000.000000
50%	76882.000000
75%	113000.000000
max	480000.000000

dtype: float64

```
df.head()
```

	currency	experience_years	year	age_num	industry	job_title	city	state	country_ml	experience_years_numeric	annua
0	USD	11 - 20 years	2019	39.5	government	other	nashville	tn	None	15.5	
1	USD	8 - 10 years	2019	29.5	other	operations director	madison	wi	None	9.0	
2	USD	2 - 4 years	2019	21.0	other	other	las vegas	nv	None	3.0	
3	GBP	5-7 years	2019	29.5	other	senior scientist	cardiff	uk	None	NaN	
4	USD	5-7 years	2019	29.5	healthcare	other	southeast michigan	usa	None	NaN	

Next steps: Generate code with df | New interactive sheet

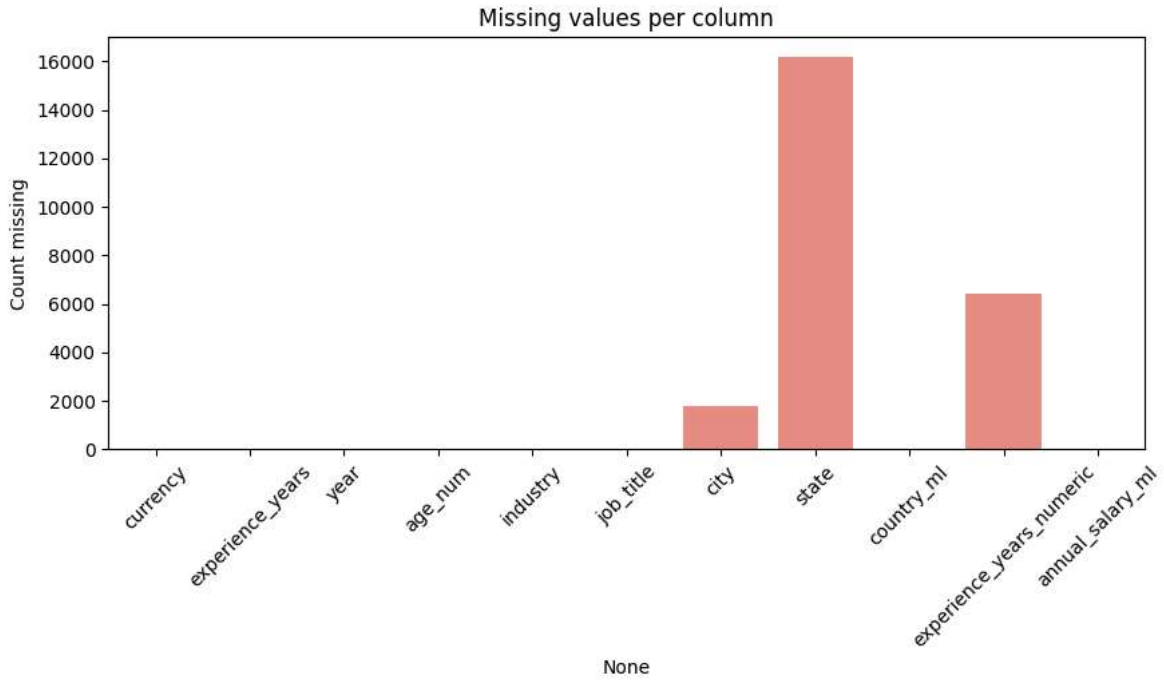
Exploratory Data Analysis

```
##First we get the overview ofthe data
# quick checks
df.shape
df.dtypes
df.isna().sum()
```

```
df.describe(include='all').T
```

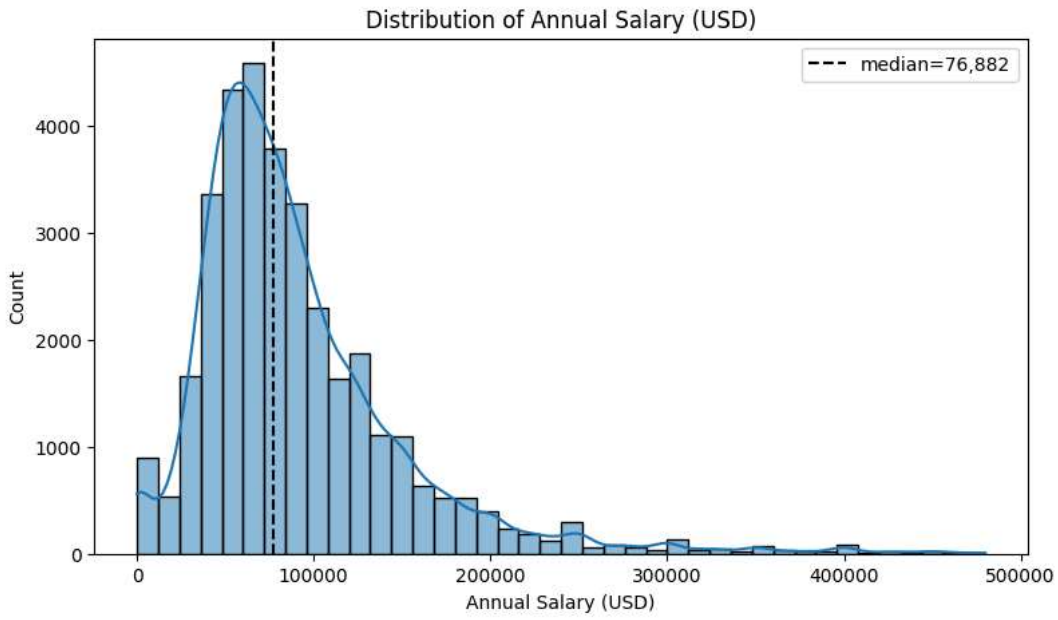
	count	unique	top	freq	mean	std	min	25%	50%	75%	max
currency	34197	6	USD	30073	NaN	NaN	NaN	NaN	NaN	NaN	NaN
experience_years	34197	8	11 - 20 years	9332	NaN	NaN	NaN	NaN	NaN	NaN	NaN
year	34197.0	NaN	NaN	NaN	2019.117349	0.521726	2019.0	2019.0	2019.0	2019.0	2025.0
age_num	34197.0	NaN	NaN	NaN	36.590827	9.946716	17.0	29.5	29.5	39.5	70.0
industry	34197	20	other	17910	NaN	NaN	NaN	NaN	NaN	NaN	NaN
job_title	34194	440	other	18299	NaN	NaN	NaN	NaN	NaN	NaN	NaN
city	32418	5981	new york	1266	NaN	NaN	NaN	NaN	NaN	NaN	NaN
state	18013	722	ca	1600	NaN	NaN	NaN	NaN	NaN	NaN	NaN
country_ml	34181	7	None	25639	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
## Dataset snapshot
### This tells us where our data is missing and what is the dataset size
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(10,4))
sns.barplot(x=df.isna().sum().index, y=df.isna().sum().values, color='salmon')
plt.xticks(rotation=45)
plt.title('Missing values per column')
plt.ylabel('Count missing')
plt.show()
```



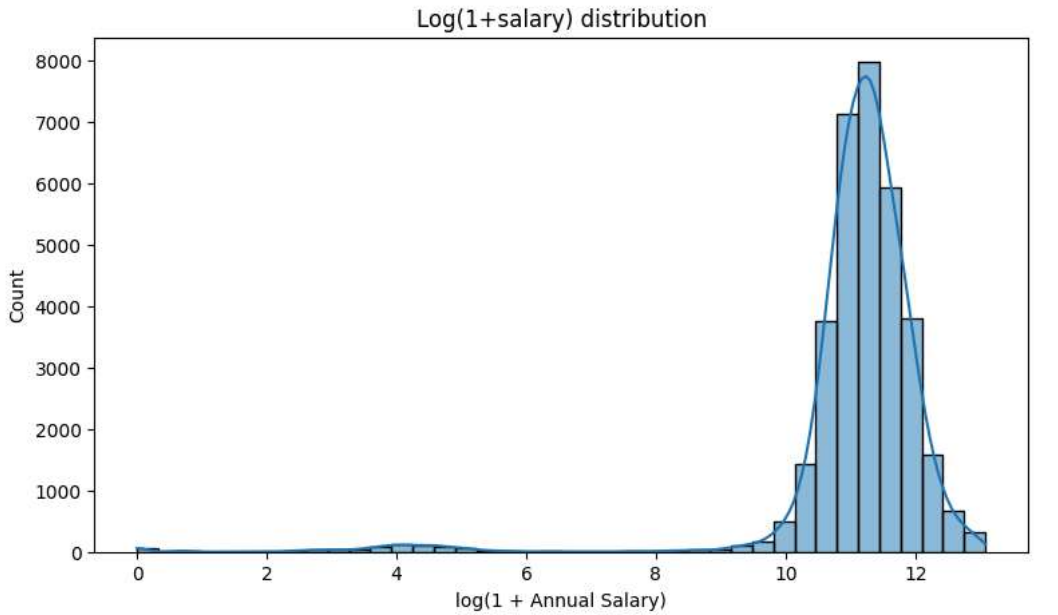
In Above plot we see state have most of the missing values so in later we decide what we do

```
plt.figure(figsize=(9,5))
sns.histplot(df['annual_salary_ml'], bins=40, kde=True)
plt.axvline(df['annual_salary_ml'].median(), color='k', linestyle='--', label=f"median={df['annual_salary_ml'].median():,.0f}")
plt.title('Distribution of Annual Salary (USD)')
plt.xlabel('Annual Salary (USD)')
plt.legend()
plt.show()
```

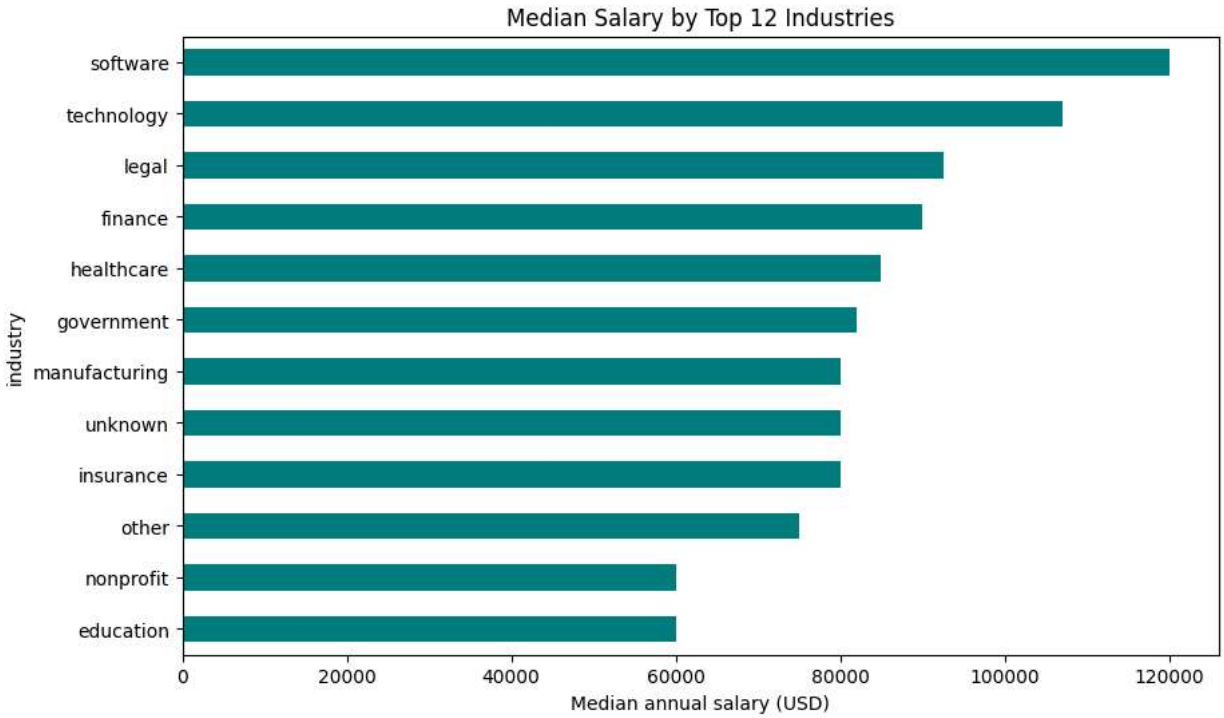


Above this graph is right skewed we make transformation to make it normalize

```
plt.figure(figsize=(9,5))
sns.histplot(np.log1p(df['annual_salary_ml']), bins=40, kde=True)
plt.title('Log(1+salary) distribution')
plt.xlabel('log(1 + Annual Salary)')
plt.show()
```



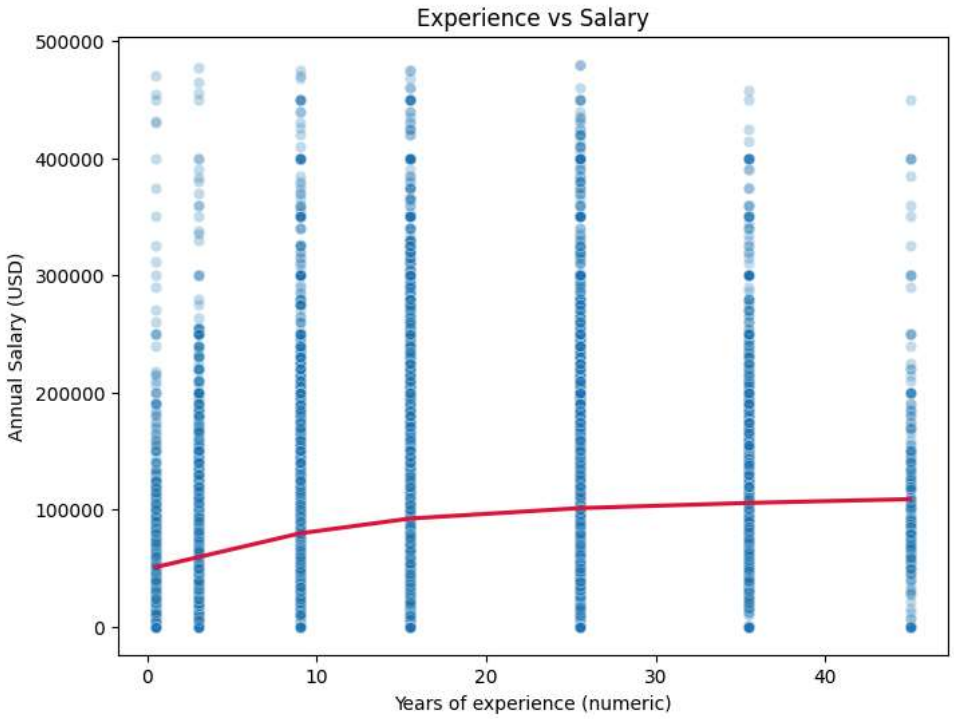

```
## Now we see median salary which gives more pay
top = df['industry'].value_counts().index[:12]
medians = df[df['industry'].isin(top)].groupby('industry')['annual_salary_ml'].median().sort_values()
plt.figure(figsize=(10,6))
medians.plot(kind='barh', color='teal')
plt.xlabel('Median annual salary (USD)')
plt.title('Median Salary by Top 12 Industries')
plt.show()
```



This shows us salary median each and we see software area have more pay as compared to others

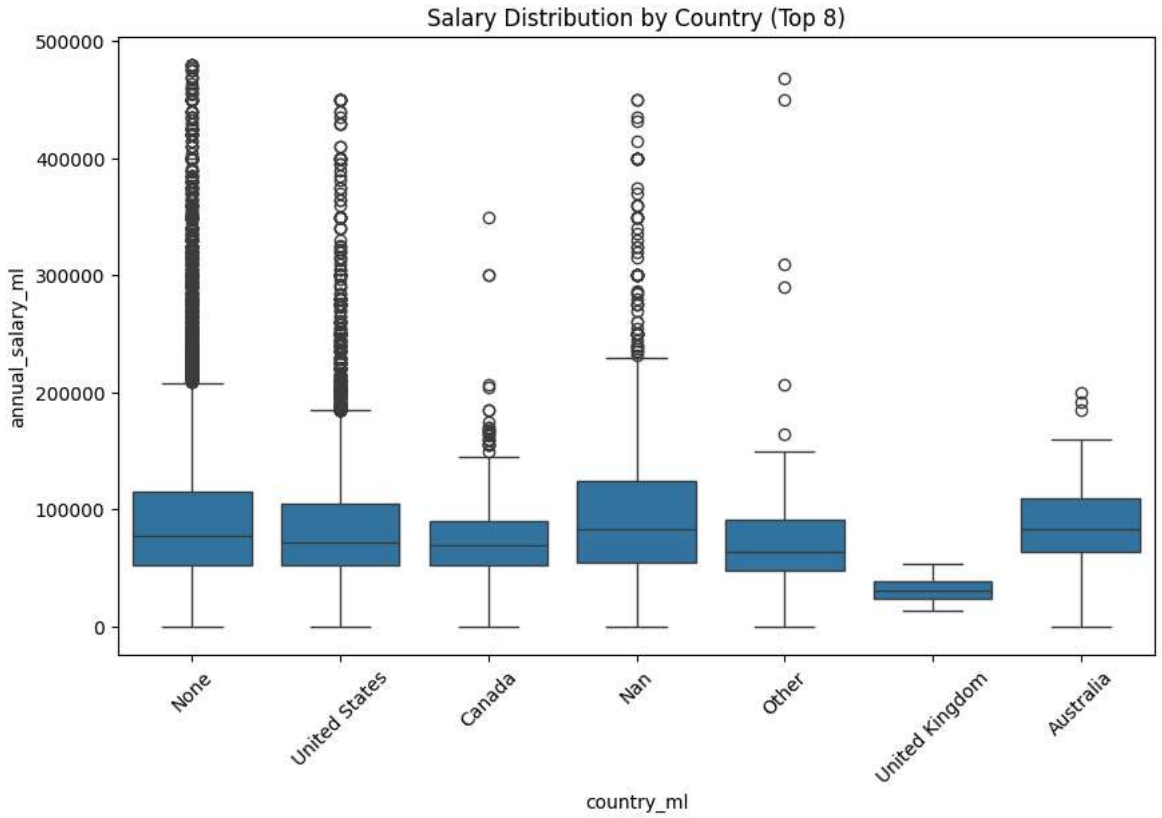
- Below we check the effect the salary based on expeience

```
plt.figure(figsize=(8,6))
sns.scatterplot(x='experience_years_numeric', y='annual_salary_ml', data=df, alpha=0.25)
sns.regplot(x='experience_years_numeric', y='annual_salary_ml', data=df, scatter=False, lowess=True, color='crimson')
plt.xlabel('Years of experience (numeric)')
plt.ylabel('Annual Salary (USD)')
plt.title('Experience vs Salary')
plt.show()
```



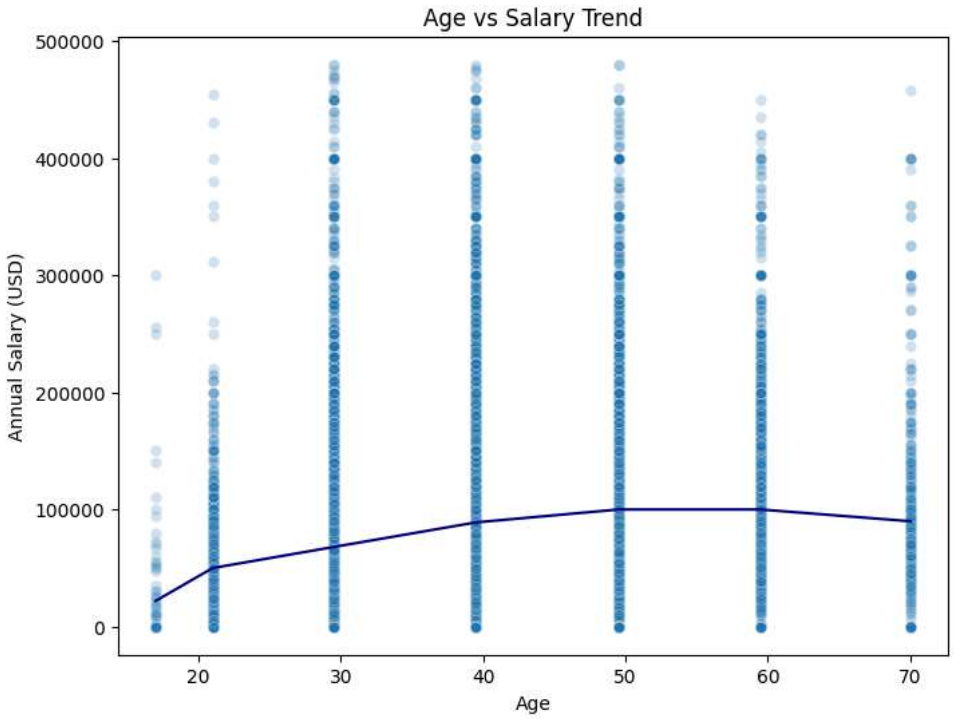
In Above graph we see there is not relying factor like we cannot say if experience is high salary must be increases companies are looking for role based

```
## After this we analyze salary based on country using box plot
top_c = df['country_ml'].value_counts().index[:8]
plt.figure(figsize=(10,6))
sns.boxplot(x='country_ml', y='annual_salary_ml', data=df[df['country_ml'].isin(top_c)])
plt.xticks(rotation=45)
plt.title('Salary Distribution by Country (Top 8)')
plt.show()
```



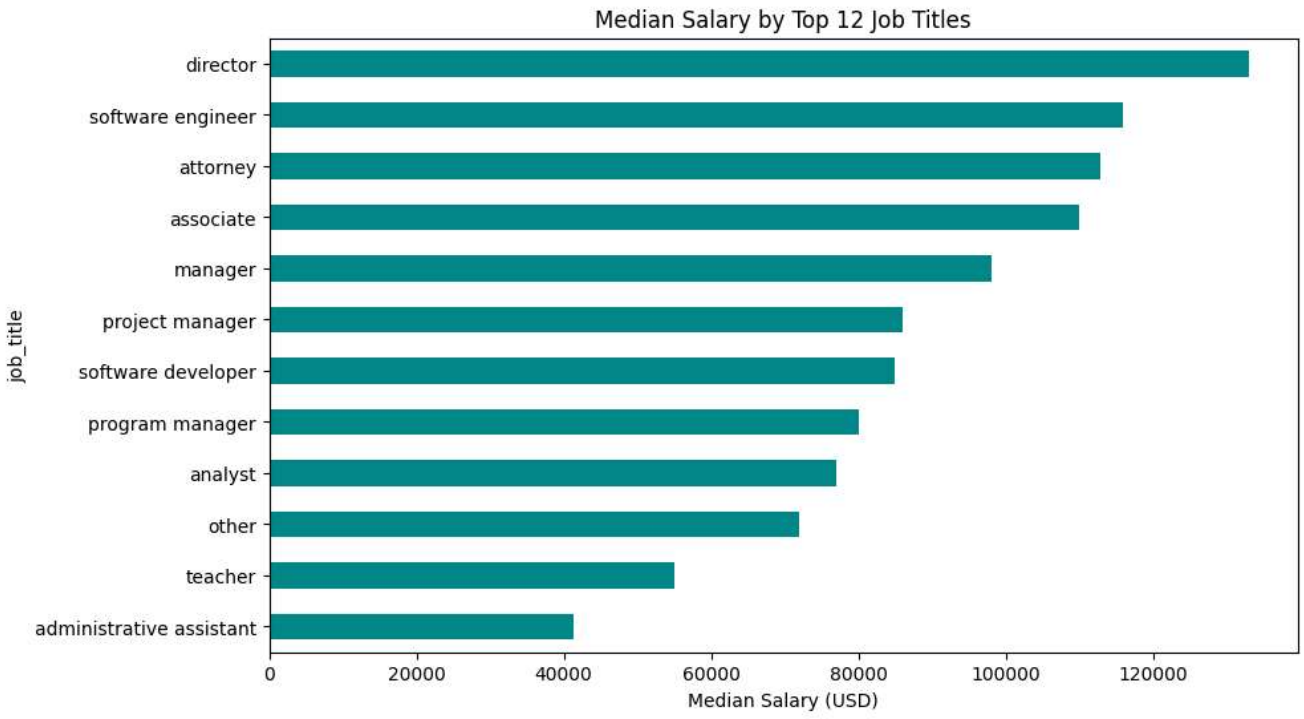
Age Vs Salary

```
plt.figure(figsize=(8,6))

# Scatter
sns.scatterplot(x='age_num', y='annual_salary_ml', data=df, alpha=0.2)

# Compute median salary grouped by age
age_median = df.groupby('age_num')['annual_salary_ml'].median().reset_index()

# Line
sns.lineplot(x='age_num', y='annual_salary_ml', data=age_median, color='navy')

plt.xlabel('Age')
plt.ylabel('Annual Salary (USD)')
plt.title('Age vs Salary Trend')
plt.show()
```



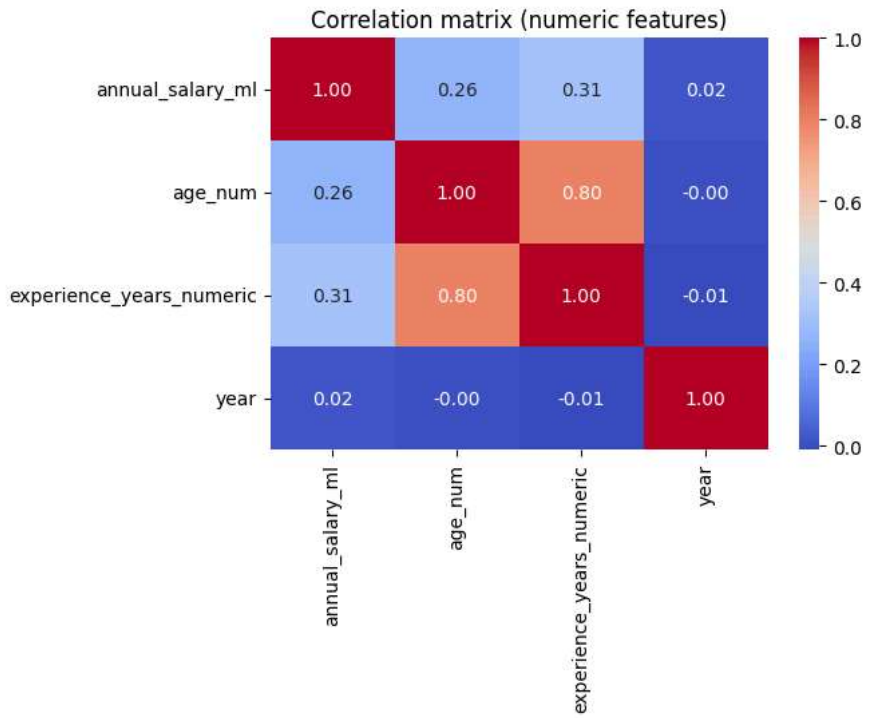
Below we finding the top 12 job titles having median salary

```
top_jobs = df['job_title'].value_counts().index[:12]
job_meds = df[df['job_title'].isin(top_jobs)].groupby('job_title')['annual_salary_ml'].median().sort_values()
plt.figure(figsize=(10,6))
job_meds.plot(kind='barh', color='darkcyan')
plt.title('Median Salary by Top 12 Job Titles')
plt.xlabel('Median Salary (USD)')
plt.show()
```




```
plt.figure(figsize=(6,4))
sns.heatmap(df[['annual_salary_ml','age_num','experience_years_numeric','year']].corr(), annot=True, fmt='.2f', cmap='coolwarm'
plt.title('Correlation matrix (numeric features)')
```

```
plt.show()
```



Correlation Matrix Interpretation

① Salary is not strongly driven by age

Older ≠ always higher salary

Skillset & job title more important than age

② Experience impacts salary — but weakly

Correlation 0.31 suggests: → Even people with low experience sometimes earn high salaries → This mostly happens in Tech / AI roles

So career field influences salary more than experience alone.

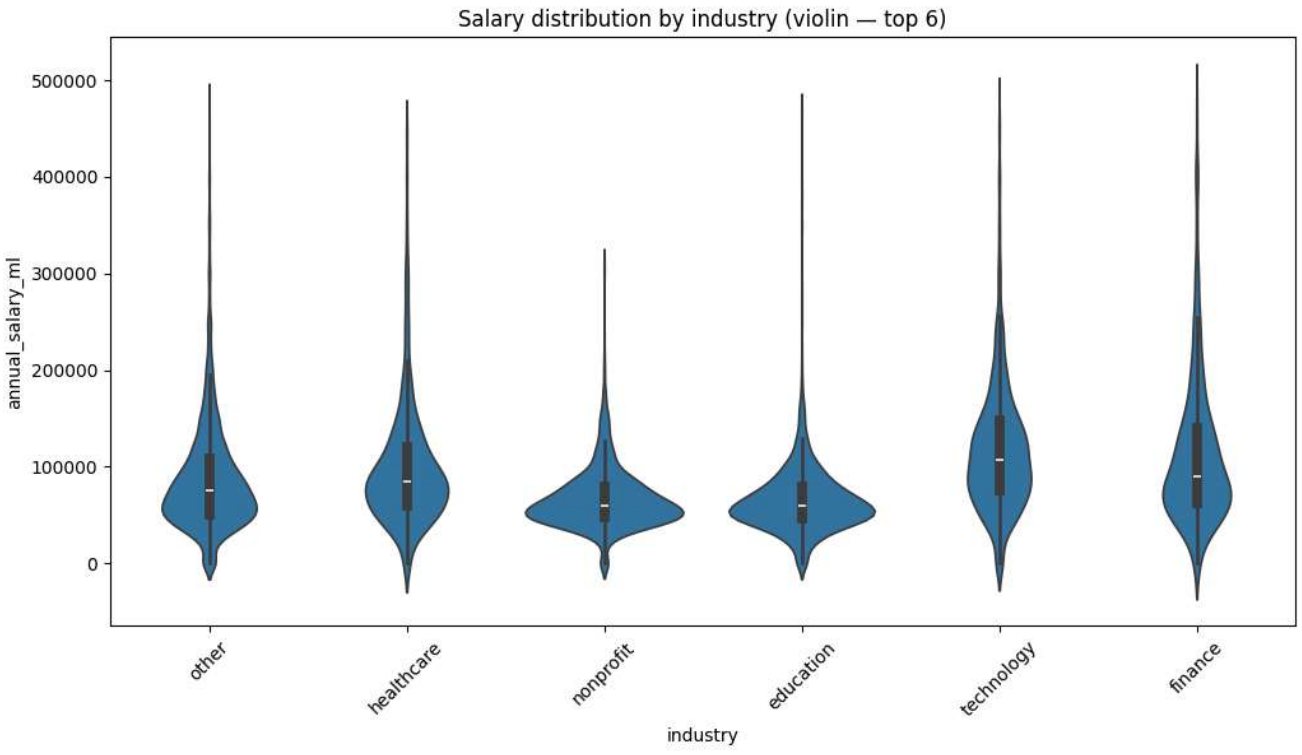
③ Age and Experience are strongly correlated

No need to use both together in a model → Multicollinearity problem

In ML: drop one (better: keep experience_years_numeric)

Now we check how experience effect salary by different indstries

```
top_ind = df['industry'].value_counts().index[:6]
plt.figure(figsize=(12,6))
sns.violinplot(x='industry', y='annual_salary_ml', data=df[df['industry'].isin(top_ind)])
plt.xticks(rotation=45)
plt.title('Salary distribution by industry (violin – top 6)')
sns.set_style("whitegrid")
sns.set_palette("Set2")
plt.rcParams['figure.dpi'] = 120
plt.show()
```



Insights We Can Fetch

🔷 1. Technology & Finance are the highest-paying industries

Their median (white dot) is higher than others.

Also, distribution is wide, meaning: → there are both entry-level and high-executive earners in these sectors.

🔷 2. Education & Nonprofit industries pay the least

Low median salaries, narrow distributions.

Suggests limited earning growth — most salaries lie in a low range.

🔷 3. Healthcare offers stable income

Median higher than Education but lower than Technology/Finance.

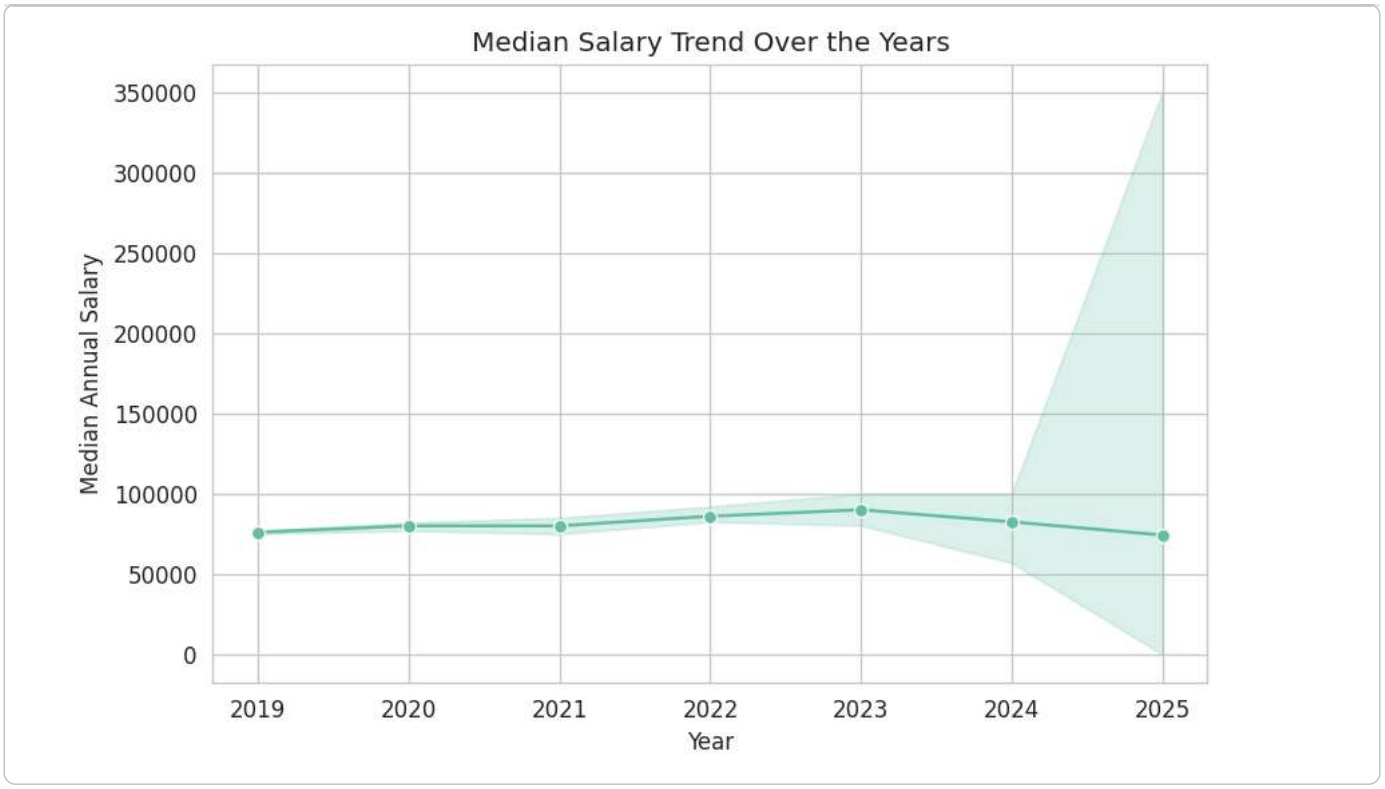
Moderate spread shows predictable pay levels (less volatility).

🔷 4. "Other" category has high variation

Indicates this group mixes many roles — from low-level to top managerial or tech-linked jobs.

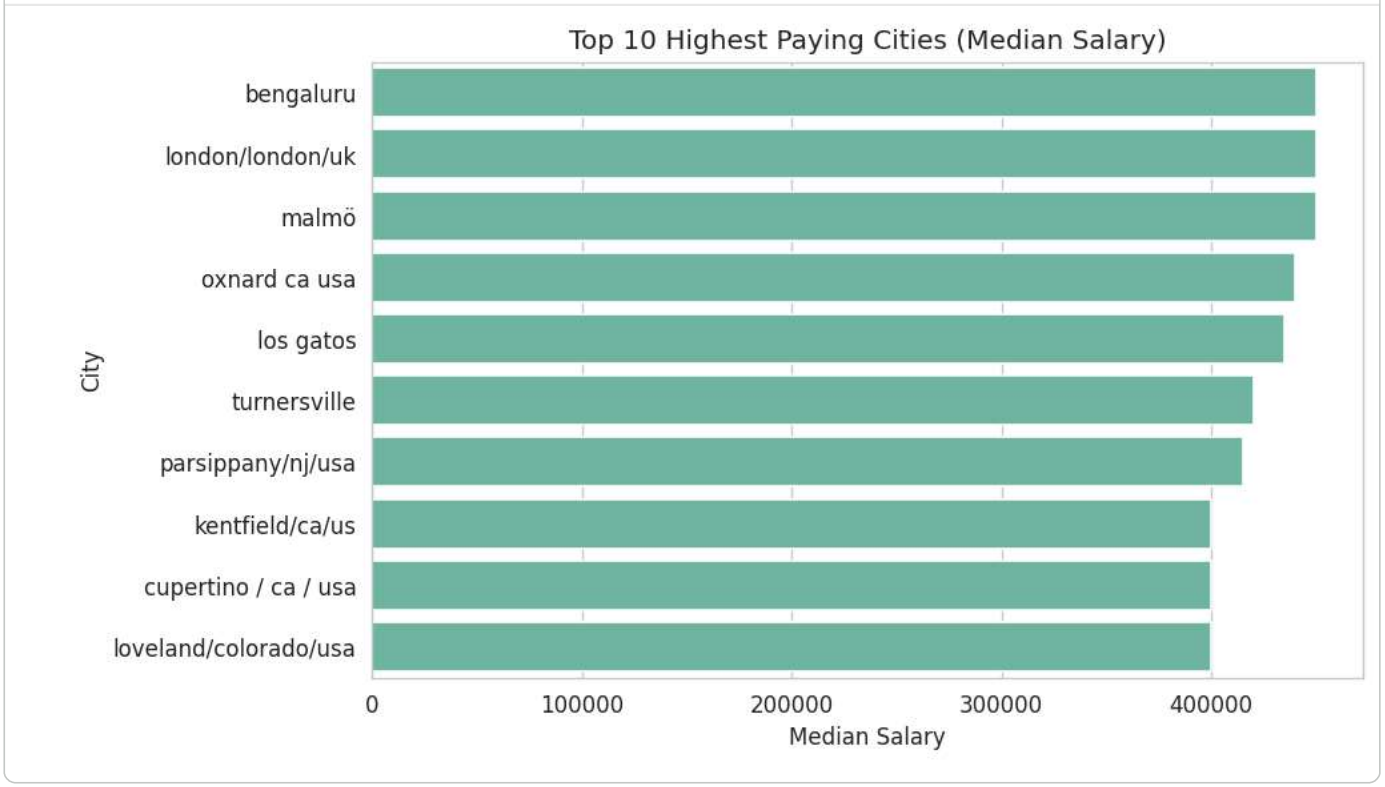
Now we see how salary effect on time

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(8,5))
sns.lineplot(x="year", y="annual_salary_ml", data=df, estimator="median", marker="o")
plt.title("Median Salary Trend Over the Years")
plt.xlabel("Year")
plt.ylabel("Median Annual Salary")
plt.grid(True)
plt.show()
```

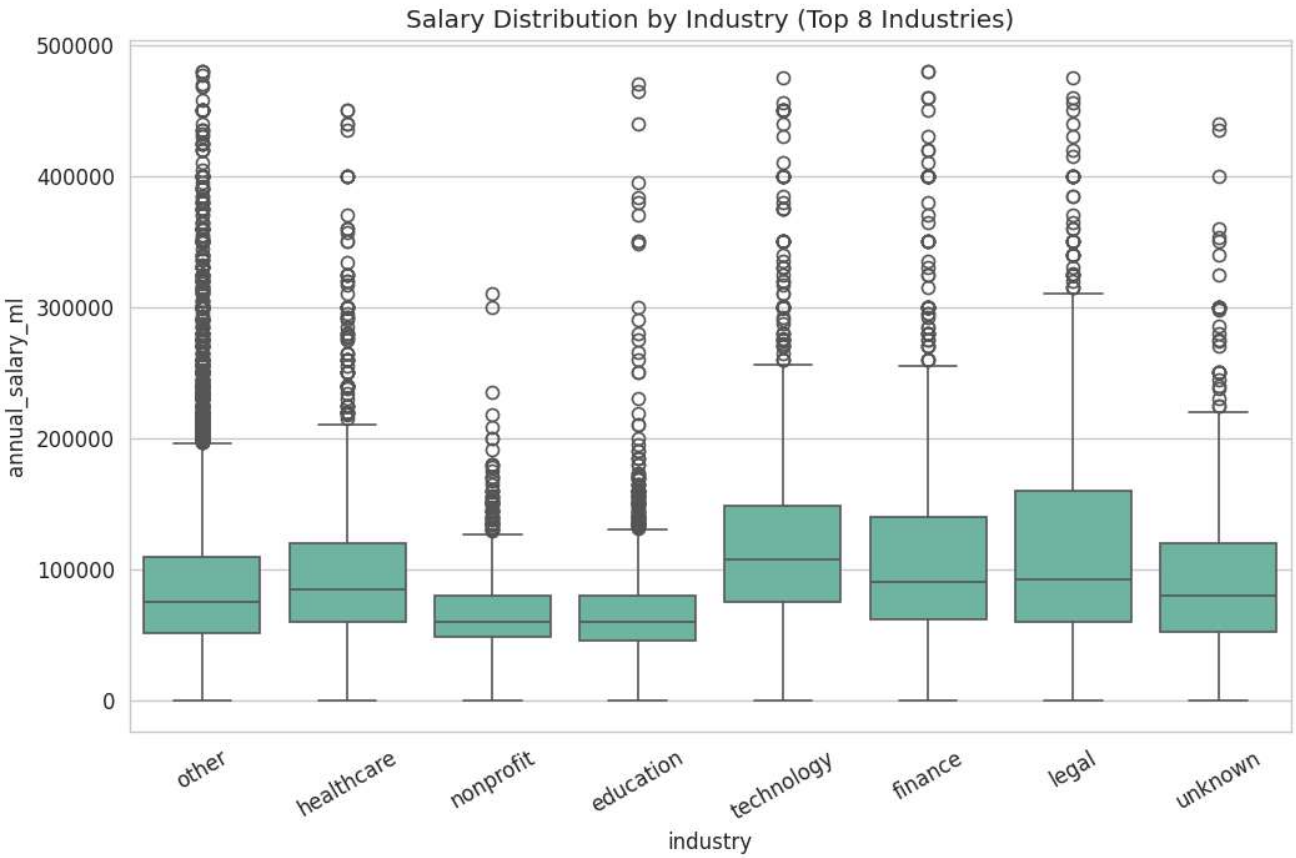


Top 10 Cities with Highest Median Salary

```
city_salary = df.groupby("city")["annual_salary_ml"].median().sort_values(ascending=False).head(10)

plt.figure(figsize=(8,5))
sns.barplot(x=city_salary.values, y=city_salary.index)
plt.title("Top 10 Highest Paying Cities (Median Salary)")
plt.xlabel("Median Salary")
plt.ylabel("City")
plt.show()
```



```
top_industries = df['industry'].value_counts().head(8).index
plt.figure(figsize=(10,6))
sns.boxplot(x="industry", y="annual_salary_ml", data=df[df['industry'].isin(top_industries)])
plt.title("Salary Distribution by Industry (Top 8 Industries)")
plt.xticks(rotation=30)
plt.show()
```



As We See State Column Having Null Values So We Statisticaly Check The Nature

```
df['state'].isna().mean()
```

```
np.float64(0.47325788811884084)
```

```
df.groupby('state')['annual_salary_ml'].median().sort_values().head(10)
df.groupby('state')['annual_salary_ml'].median().sort_values().tail(10)
```

state	annual_salary_ml
pa/usa	277000.0
in- cover in	290000.0
new york. usa	290000.0
ane	300000.0
60613	350000.0
georga	350000.0
czech republik	360000.0
ea	400000.0
skåne	450000.0
stockholms län	468000.0

dtype: float64

```
from scipy.stats import f_oneway

groups = [group['annual_salary_ml'].dropna().values
          for name, group in df.groupby('state') if group['annual_salary_ml'].notna().sum() > 30]

f_stat, p_value = f_oneway(*groups)
p_value
```

```
np.float64(1.1953676298147019e-201)
```

Based on above analysis we conclude that State Column does not have a strong impact on our case like nearly 50% values are missing we anova reslts are also too low

```
## Dropping state column
df.drop(columns=['state'], inplace=True)
```

```
df.head()
```

	currency	experience_years	year	age_num	industry	job_title	city	country_ml	experience_years_numeric	annual_salar
0	USD	11 - 20 years	2019	39.5	government	other	nashville	None	15.5	750
1	USD	8 - 10 years	2019	29.5	other	operations director	madison	None	9.0	650
2	USD	2 - 4 years	2019	21.0	other	other	las vegas	None	3.0	363
3	GBP	5-7 years	2019	29.5	other	senior scientist	cardiff	None	NaN	346
4	USD	5-7 years	2019	29.5	healthcare	other	southeast	None	NaN	550

Next steps: Generate code with df | New interactive sheet

```
df['currency'].unique()
```

```
array(['USD', 'GBP', 'CAD', 'EUR', 'AUD', 'OTHER'], dtype=object)
```

```
## Creating the dictionary store the basic conversion
# Step 1: Create a rate dictionary
import numpy as np
rates_to_usd = {
    'USD': 1.00,
    'GBP': 1.27,
    'EUR': 1.08,
    'CAD': 0.74,
    'AUD': 0.66,
    'OTHER': np.nan
}
```

```
# Step 2: Convert to USD
df['salary_usd'] = df.apply(
    lambda row: row['annual_salary_ml'] * rates_to_usd.get(row['currency'], np.nan),
    axis=1
)

# Step 3: Quick check
df[['currency', 'annual_salary_ml', 'salary_usd']].head()
```

	currency	annual_salary_ml	salary_usd
0	USD	75000.0	75000.0
1	USD	65000.0	65000.0
2	USD	36330.0	36330.0
3	GBP	34600.0	43942.0
4	USD	55000.0	55000.0

```
df.head()
```

	currency	experience_years	year	age_num	industry	job_title	city	country_ml	experience_years_numeric	annual_salar
0	USD	11 - 20 years	2019	39.5	government	other	nashville	None	15.5	750
1	USD	8 - 10 years	2019	29.5	other	operations director	madison	None	9.0	650
2	USD	2 - 4 years	2019	21.0	other	other	las vegas	None	3.0	363
3	GBP	5-7 years	2019	29.5	other	senior scientist	cardiff	None	NaN	346
4	USD	5-7 years	2019	29.5	healthcare	other	southeast michigan	None	NaN	550

Next steps: Generate code with df | New interactive sheet

Now We Removing Outliers From all our numeric columns

```
numeric_cols=['age_num','experience_years_numeric','annual_salary_ml','salary_usd']
## Make the copy of original data
df_clean=df.copy()
for col in numeric_cols:
  lower=df_clean[col].quantile(0.01)
  upper=df_clean[col].quantile(0.99)
  df_clean=df_clean[(df_clean[col]>=lower)&(df_clean[col]<=upper)]

print(f"Remaining rows: {df_clean.shape[0]}")
print(f"Removed rows: {df.shape[0] - df_clean.shape[0]}")
```

```
Remaining rows: 26180
Removed rows: 8017
```

```
## Replace the original coulmn
df=df_clean.copy()
```

```
Start coding or generate with AI.
```

Performig EDA Again On Data Which have no outliers

```
df.head()
```

	currency	experience_years	year	age_num	industry	job_title	city	country_ml	experience_years_numeric	annual_salar
0	USD	11 - 20 years	2019	39.5	government	other	nashville	None	15.5	75
1	USD	8 - 10 years	2019	29.5	other	operations director	madison	None	9.0	65
2	USD	2 - 4 years	2019	21.0	other	other	las vegas	None	3.0	36
5	USD	8 - 10 years	2019	29.5	other	other	seattle	None	9.0	45
6	USD	2 - 4 years	2019	29.5	nonprofit	development manager	dallas	United States	3.0	51

Next steps: Generate code with df New interactive sheet

```
df.head()
```

	currency	experience_years	year	age_num	industry	job_title	city	country_ml	experience_years_numeric	annual_salar
0	USD	11 - 20 years	2019	39.5	government	other	nashville	None	15.5	75
1	USD	8 - 10 years	2019	29.5	other	operations director	madison	None	9.0	65
2	USD	2 - 4 years	2019	21.0	other	other	las vegas	None	3.0	36
5	USD	8 - 10 years	2019	29.5	other	other	seattle	None	9.0	45
6	USD	2 - 4 years	2019	29.5	nonprofit	development manager	dallas	United States	3.0	51

Next steps: Generate code with df New interactive sheet

```
df['salary_final'] = np.sqrt(df['salary_usd'])
```

```
df.drop(columns=['salary_usd','annual_salary_ml','year','experience_years','currency'],inplace=True)
```

```
df.head()
```

	age_num	industry	job_title	city	country_ml	experience_years_numeric	salary_final
0	39.5	government	other	nashville	None	15.5	273.861279
1	29.5	other	operations director	madison	None	9.0	254.950976
2	21.0	other	other	las vegas	None	3.0	190.604302
5	29.5	other	other	seattle	None	9.0	212.132034
6	29.5	nonprofit	development manager	dallas	United States	3.0	225.831796

Next steps: Generate code with df New interactive sheet

```
df.isnull().sum()
```

	0
age_num	0
industry	0
job_title	1
city	1309
country_ml	11
experience_years_numeric	0
salary_final	0

dtype: int64

```
df.describe()
```

	age_num	experience_years_numeric	salary_final
count	26180.000000	26180.000000	26180.000000
mean	37.383613	12.802139	288.046882
std	9.795703	8.779897	78.915226
min	21.000000	0.500000	14.142136
25%	29.500000	3.000000	231.188235
50%	39.500000	15.500000	279.284801
75%	39.500000	15.500000	339.116499
max	59.500000	35.500000	527.257053

Based On Our Analysis We remove City column because it have high cardinalty and not too much useful for our analysis

```
df.head()
```

	age_num	industry	job_title	city	country_ml	experience_years_numeric	salary_final
0	39.5	government	other	nashville	None	15.5	273.861279
1	29.5	other	operations director	madison	None	9.0	254.950976
2	21.0	other	other	las vegas	None	3.0	190.604302
5	29.5	other	other	seattle	None	9.0	212.132034
6	29.5	nonprofit	development manager	dallas	United States	3.0	225.831796

Next steps: Generate code with df | New interactive sheet

```
##Removing City column
df.drop(columns=['city'],inplace=True)
```

```
df.head()
```

	age_num	industry	job_title	country_ml	experience_years_numeric	salary_final
0	39.5	government	other	None	15.5	273.861279
1	29.5	other	operations director	None	9.0	254.950976
2	21.0	other	other	None	3.0	190.604302
5	29.5	other	other	None	9.0	212.132034
6	29.5	nonprofit	development manager	United States	3.0	225.831796

Next steps: Generate code with df | New interactive sheet

```
## Replace with unknown in job title column
df['job_title'].fillna("Unknown", inplace=True)
```

```
/tmp/ipython-input-4233705557.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chaine
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are sett

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df


  df['job_title'].fillna("Unknown", inplace=True)
```

```
## Same with country_ml
df['country_ml'].fillna("Unknown",inplace=True)
```

```
/tmp/ipython-input-2575076523.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chaine
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are sett
```

```
For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df
```

```
df['country_ml'].fillna("Unknown",inplace=True)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 26180 entries, 0 to 35478
Data columns (total 6 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   age_num                   26180 non-null  float64
 1   industry                  26180 non-null  object 
 2   job_title                 26180 non-null  object 
 3   country_ml                26180 non-null  object 
 4   experience_years_numeric  26180 non-null  float64
 5   salary_final              26180 non-null  float64
dtypes: float64(3), object(3)
memory usage: 1.4+ MB
```

```
df.head(100)
```

	age_num	industry	job_title	country_ml	experience_years_numeric	salary_final
0	39.5	government	other	None	15.5	273.861279
1	29.5	other	operations director	None	9.0	254.950976
2	21.0	other	other	None	3.0	190.604302
5	29.5	other	other	None	9.0	212.132034
6	29.5	nonprofit	development manager	United States	3.0	225.831796
...	...	...	...	...	...	...
126	39.5	construction	other	None	15.5	269.072481
127	29.5	education	other	None	15.5	192.353841
128	39.5	healthcare	other	None	15.5	374.165739
129	29.5	other	property manager	United States	9.0	223.606798
130	49.5	other	other	None	25.5	316.227766

100 rows × 6 columns

Next steps: Generate code with df New interactive sheet

```
df['country_ml'].unique()
```

```
array(['None', 'United States', 'Canada', 'Nan', 'Other', 'Australia',
       'United Kingdom', 'Unknown'], dtype=object)
```

```
df.head()
```

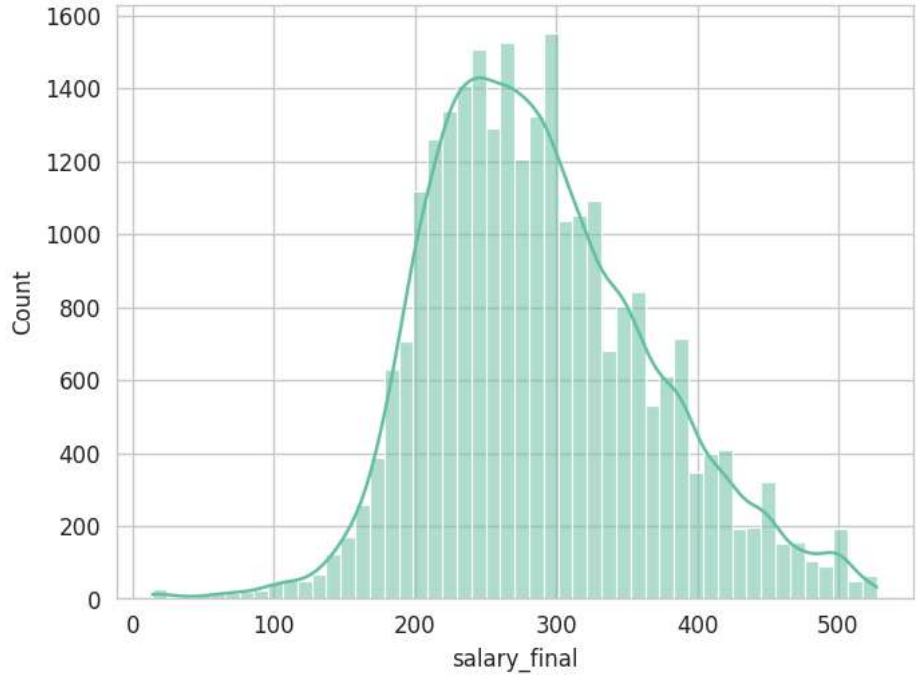
	age_num	industry	job_title	country_ml	experience_years_numeric	salary_final
0	39.5	government	other	None	15.5	273.861279
1	29.5	other	operations director	None	9.0	254.950976
2	21.0	other	other	None	3.0	190.604302
5	29.5	other	other	None	9.0	212.132034
6	29.5	nonprofit	development manager	United States	3.0	225.831796

Next steps: Generate code with df New interactive sheet

```
# basic
print(df.shape)
print(df.dtypes)
print(df.isna().sum())

# target quick stats
print(df['salary_final'].describe())
```

```
import seaborn as sns, matplotlib.pyplot as plt
sns.histplot(df['salary_final'], bins=50, kde=True); plt.show()
```

```
(26180, 6)
age_num                     float64
industry                     object
job_title                    object
country_ml                   object
experience_years_numeric    float64
salary_final                float64
dtype: object
age_num                     0
industry                    0
job_title                   0
country_ml                  0
experience_years_numeric    0
salary_final                0
dtype: int64
count    26180.000000
mean       288.046882
std         78.915226
min         14.142136
25%        231.188235
50%        279.284801
75%        339.116499
max        527.257053
Name: salary_final, dtype: float64
```




```
df['job_title'].unique()
```

```
       'supervisor', 'development officer', 'content manager',
       'audit manager', 'database administrator', 'financial advisor',
       'intern', 'corporate counsel', 'senior vice president',
       'sr financial analyst', 'research coordinator',
       'assistant store manager', 'phd student', 'senior director',
       'clinical psychologist', 'associate product manager', 'qa analyst',
       'chief financial officer', 'marketing specialist',
       'engagement manager', 'buyer', 'compliance officer',
       'marketing analyst', 'director of product management', 'ceo',
       'electrical engineer', 'professor', 'auditor', 'analytics manager',
       'sales engineer', 'product designer', 'it project manager', 'cto',
       'technical director', 'associate manager', 'process engineer',
       'loan officer', 'principal consultant', 'police officer',
       'physician assistant', 'director of engineering', 'optometrist',
       'administrative officer', 'school psychologist',
       'postdoctoral fellow', 'assistant vice president', 'sr engineer',
       'director of sales', 'server', 'training specialist',
       'occupational therapist', 'operations analyst',
       'system administrator', 'business development manager',
       'sales director', 'staff engineer', 'marketing', 'it specialist',
       'sales representative', 'credit analyst',
       'principal software engineer', 'asset manager', 'sales executive',
       'assistant controller', 'doctor', 'front end developer',
       'psychologist', 'legal counsel', 'industrial engineer',
       'electrician', 'sales', 'cashier', 'policy advisor',
       'security analyst', 'information security analyst',
       'solution architect', 'chief operating officer',
       'software development manager', 'devops engineer', 'nurse',
       'site reliability engineer', 'pilot', 'sr software engineer',
       'logistics manager', 'interior designer', 'regional sales manager',
       'cio', 'dentist', 'driver', 'technology manager', 'sales rep',
       'site manager', 'sr program manager', 'trader', 'Unknown',
       'sales analyst', 'purchasing manager', 'system engineer', 'svp',
       'procurement manager', 'chief engineer', 'director of it',
       'superintendent', 'sr product manager', 'network administrator',
       'md'], dtype=object)
```

```
# === FULL FEATURE ENGINEERING PIPELINE (run on your df) ===

import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from sklearn.preprocessing import StandardScaler

# ---------- 0. Quick safety checks ----------
assert 'job_title' in df.columns, "DataFrame must contain 'job_title' column."
print("Initial shape:", df.shape)

# Keep a working copy
df_fe = df.copy()

# ---------- 1. Ensure cleaned title exists ----------
if 'job_title_clean' not in df_fe.columns:
    df_fe['job_title_clean'] = df_fe['job_title'].astype(str).str.lower().str.strip()
    df_fe['job_title_clean'] = df_fe['job_title_clean'].str.replace(r'[_\-/\(\)\.]', ' ', regex=True)
    df_fe['job_title_clean'] = df_fe['job_title_clean'].str.replace(r'\s+', ' ', regex=True).str.strip()

# ---------- 2. Improved job_family mapping (reduce "Other") ----------
def improved_job_family(t):
    if pd.isna(t): return 'Other'
    s = t.lower()
    # Tech / IT
    tech_kw = ['software','developer','devops','engineer','programmer','backend','frontend','sde','qa','site reliability','sysad
    if any(k in s for k in tech_kw): return 'Tech / IT'
    # Data & Analytics
    data_kw = ['data ','data scientist','data analyst','machine learning','ml ','ai ','analyst','bi ','business intelligence','s
    if any(k in s for k in data_kw): return 'Data & Analytics'
```