In [2]:
```
Internship Project Topic:TCS iON RIO-125: HR Salary Dashboard - Train the Dataset and Predict Salary

Name of the Organization:TCS iON

Name of the Industry Mentor:Debashis Roy

Name of the Institute:B. K. Birla College of Arts, Science & Commerce (Autonomous), Kalyan
```

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [4]:
```
df = pd.read_csv("ds_salaries.csv")
```

In [5]:
```
df
```

Out[5]:

	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_residence	remote_ratio	company_location	c
0	2023	SE	FT	Principal Data Scientist	80000	EUR	85847	ES	100	ES	
1	2023	MI	CT	ML Engineer	30000	USD	30000	US	100	US	
2	2023	MI	CT	ML Engineer	25500	USD	25500	US	100	US	
3	2023	SE	FT	Data Scientist	175000	USD	175000	CA	100	CA	
4	2023	SE	FT	Data Scientist	120000	USD	120000	CA	100	CA	
...	...	...	...	...	...	...	...	...	...	...	
3750	2020	SE	FT	Data Scientist	412000	USD	412000	US	100	US	
3751	2021	MI	FT	Principal Data Scientist	151000	USD	151000	US	100	US	
3752	2020	EN	FT	Data Scientist	105000	USD	105000	US	100	US	
3753	2020	EN	CT	Business Data Analyst	100000	USD	100000	US	100	US	
3754	2021	SE	FT	Data Science Manager	7000000	INR	94665	IN	50	IN	

3755 rows × 11 columns

In [6]:
```
df.head(10)
```

Out[6]:

	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_residence	remote_ratio	company_location	comp
0	2023	SE	FT	Principal Data Scientist	80000	EUR	85847	ES	100	ES	
1	2023	MI	CT	ML Engineer	30000	USD	30000	US	100	US	
2	2023	MI	CT	ML Engineer	25500	USD	25500	US	100	US	
3	2023	SE	FT	Data Scientist	175000	USD	175000	CA	100	CA	
4	2023	SE	FT	Data Scientist	120000	USD	120000	CA	100	CA	
5	2023	SE	FT	Applied Scientist	222200	USD	222200	US	0	US	
6	2023	SE	FT	Applied Scientist	136000	USD	136000	US	0	US	
7	2023	SE	FT	Data Scientist	219000	USD	219000	CA	0	CA	
8	2023	SE	FT	Data Scientist	141000	USD	141000	CA	0	CA	
9	2023	SE	FT	Data Scientist	147100	USD	147100	US	0	US	

In [ ]:
```
Exploratory Data Analysis (EDA):
```

In [7]:
```
df.shape
```

Out[7]:
```
(3755, 11)
```

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3755 entries, 0 to 3754
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   work_year           3755 non-null   int64 
 1   experience_level    3755 non-null   object
 2   employment_type     3755 non-null   object
 3   job_title           3755 non-null   object
 4   salary              3755 non-null   int64 
 5   salary_currency     3755 non-null   object
 6   salary_in_usd       3755 non-null   int64 
 7   employee_residence  3755 non-null   object
 8   remote_ratio        3755 non-null   int64 
 9   company_location    3755 non-null   object
 10  company_size        3755 non-null   object
dtypes: int64(4), object(7)
memory usage: 322.8+ KB
```

In [9]:
```
df.columns
```

Out[9]:
```
Index(['work_year', 'experience_level', 'employment_type', 'job_title',
       'salary', 'salary_currency', 'salary_in_usd', 'employee_residence',
       'remote_ratio', 'company_location', 'company_size'],
      dtype='object')
```

In [10]:
```
df.describe()
```

Out[10]:

	work_year	salary	salary_in_usd	remote_ratio
count	3755.000000	3.755000e+03	3755.000000	3755.000000
mean	2022.373635	1.906956e+05	137570.389880	46.271638
std	0.691448	6.716765e+05	63055.625278	48.589050
min	2020.000000	6.000000e+03	5132.000000	0.000000
25%	2022.000000	1.000000e+05	95000.000000	0.000000
50%	2022.000000	1.380000e+05	135000.000000	0.000000
75%	2023.000000	1.800000e+05	175000.000000	100.000000
max	2023.000000	3.040000e+07	450000.000000	100.000000

In [11]:
```
df.dtypes
```

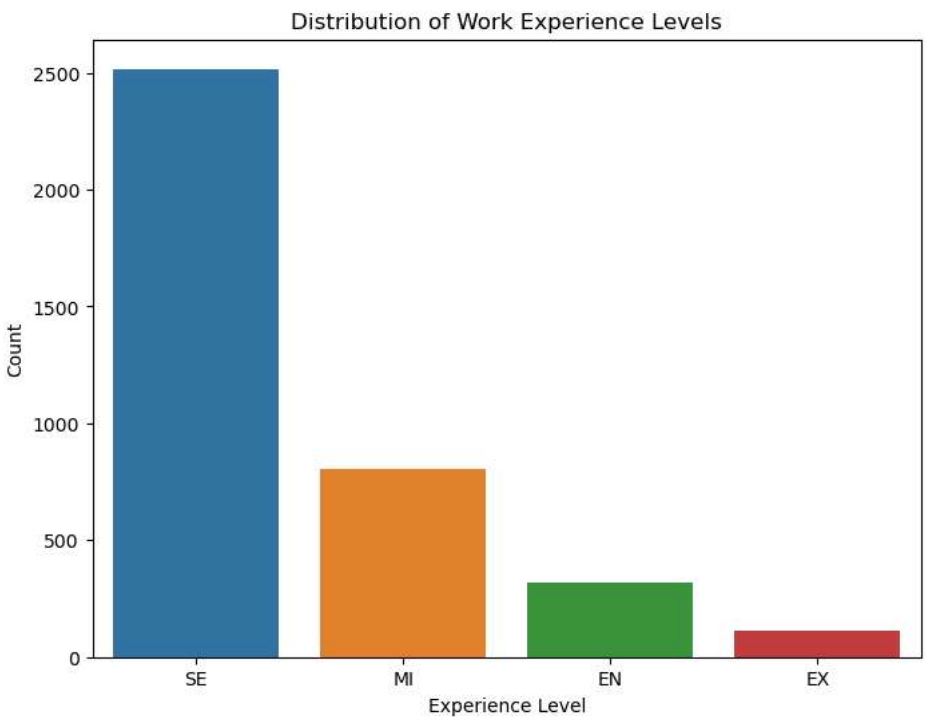
Out[11]:
```
work_year              int64
experience_level      object
employment_type       object
job_title             object
salary                 int64
salary_currency       object
salary_in_usd          int64
employee_residence    object
remote_ratio           int64
company_location      object
company_size          object
dtype: object
```

In [12]:
```
df.nunique()
```

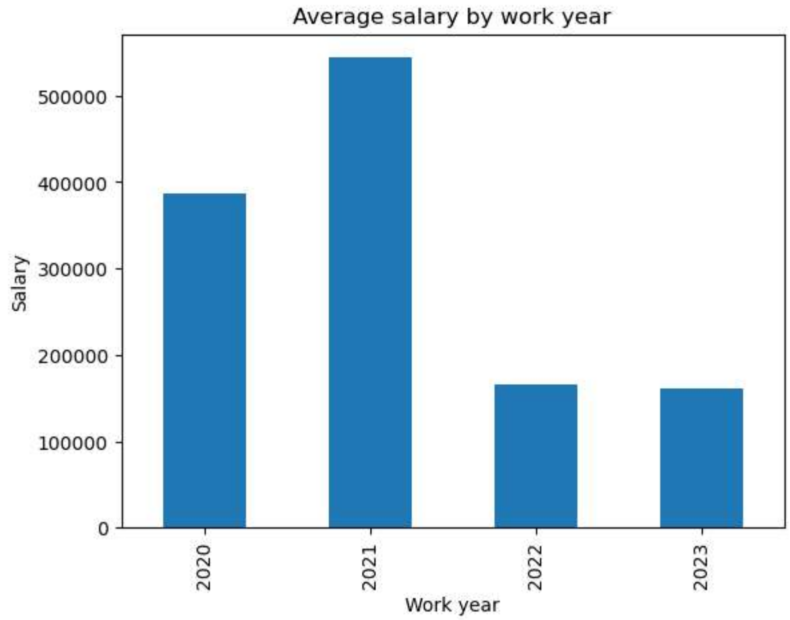
Out[12]:
```
work_year                4
experience_level         4
employment_type          4
job_title               93
salary                 815
salary_currency         20
salary_in_usd         1035
employee_residence      78
remote_ratio             3
company_location        72
company_size             3
dtype: int64
```

In [ ]:
```
There are 4 categorical values in the column "experience_level", such as:
 EN, which is Entry-level.
 MI, which is Mid-level.
 SE, which is Senior-level.
 EX, which is Executive-level.

There are 3 categorical values in the column "remote_ratio", such as:
 100, which is Remotely.
 0, which is On-site.
 50, which is Hybrid
```

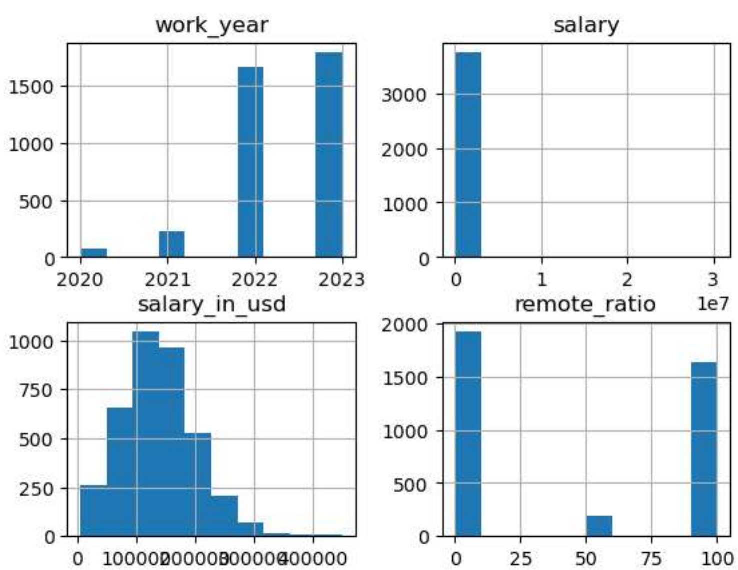
In [13]:
```
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='experience_level')
plt.xlabel('Experience Level')
plt.ylabel('Count')
plt.title('Distribution of Work Experience Levels')
plt.show()
```



In [14]:
```
df1 = df.groupby("work_year")["salary"].mean()
df1.plot(kind="bar")

plt.title("Average salary by work year")
plt.xlabel("Work year")
plt.ylabel("Salary")

plt.show()
```



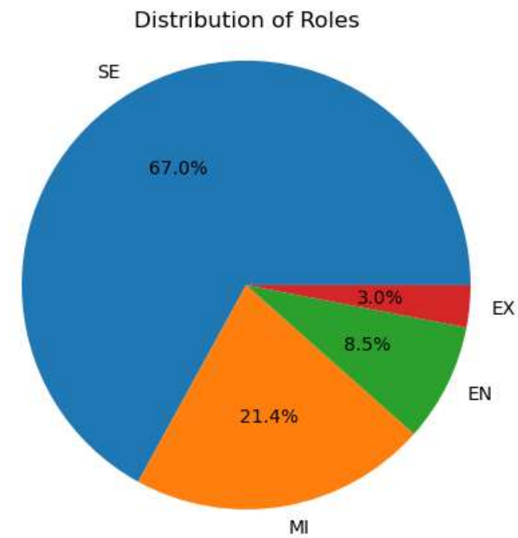
In [15]:
```
df.hist()
plt.show()
```



In [16]:
```
print("Total value counts of the roles:-\n ",df["experience_level"].value_counts())
```

```
Total value counts of the roles:-
  experience_level
SE    2516
MI     805
EN     320
EX     114
Name: count, dtype: int64
```

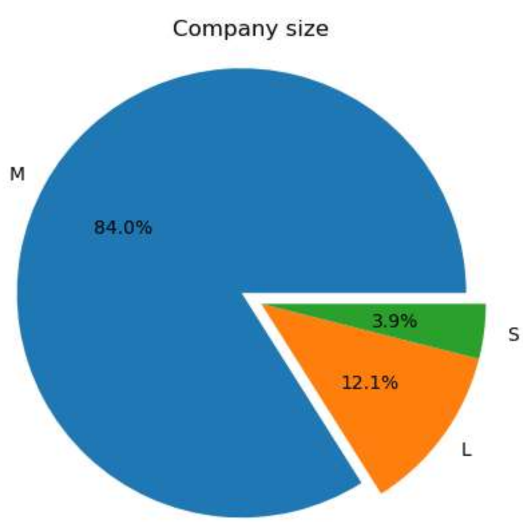
In [17]:
```
roles = ["SE", "MI", "EN", "EX"]
people = [2516, 805, 320, 114]

plt.pie(people, labels=roles, autopct='%1.1f%%')
plt.title('Distribution of Roles')
plt.axis('equal')
plt.show()
```



In [18]:
```
print(df["employment_type"].value_counts())
```

```
employment_type
FT    3718
PT      17
CT      10
FL      10
Name: count, dtype: int64
```

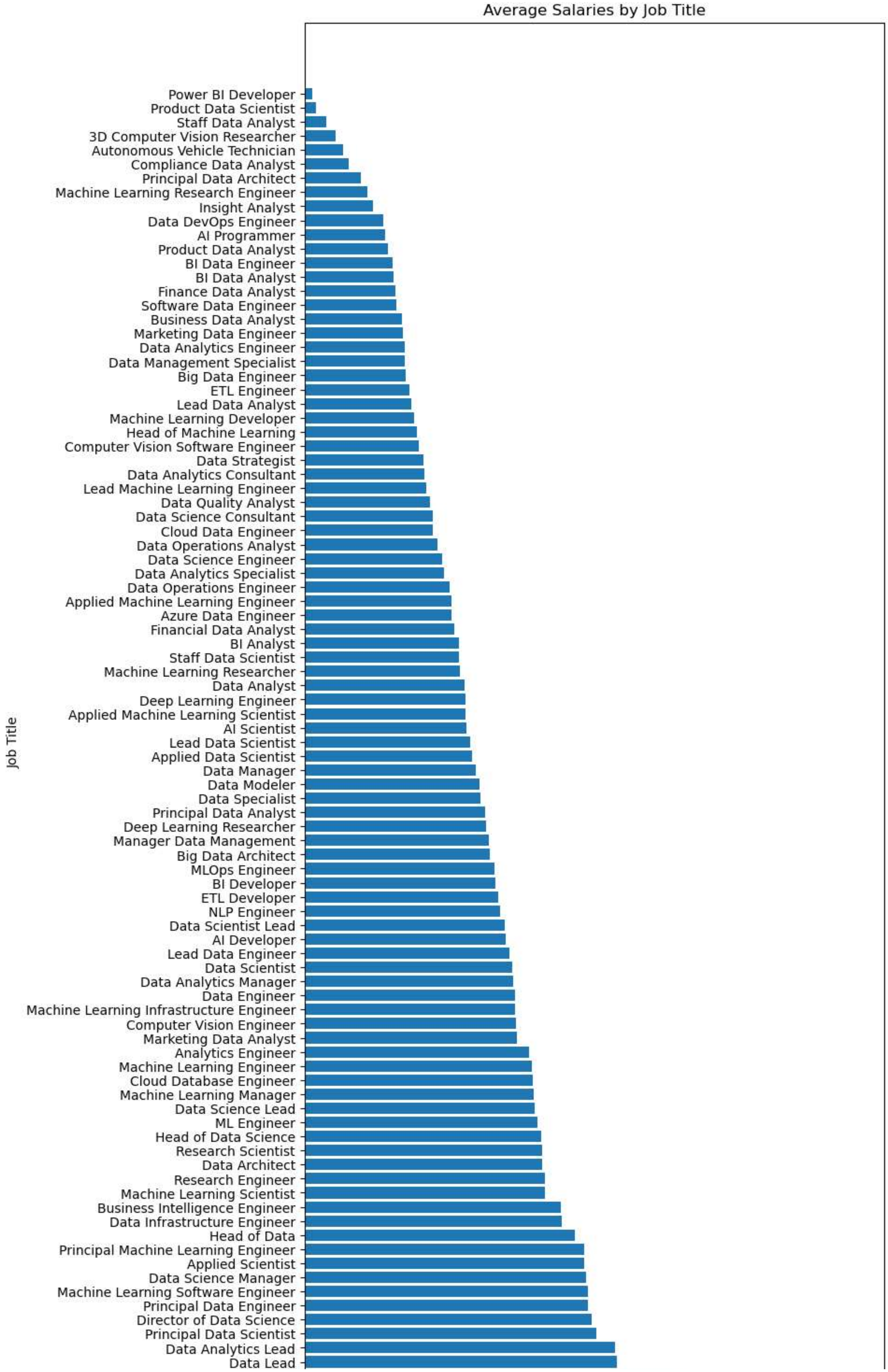
In [19]:
```
company_numbers = [3153, 454, 148]
company_size = ["M", "L", "S"]
explode = [0.1, 0, 0]
plt.pie(company_numbers, labels=company_size, explode=explode, autopct='%1.1f%%')
plt.axis('equal')
plt.title("Company size")
plt.show()
```

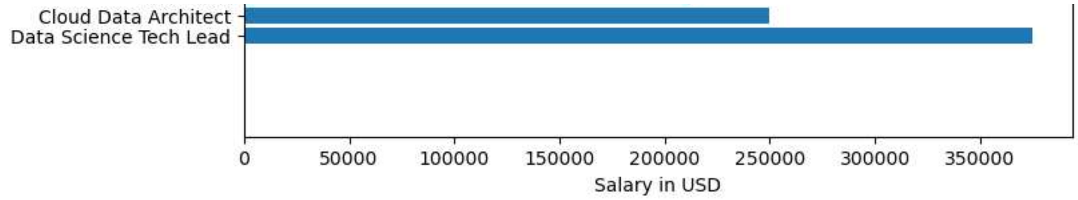


In [20]:
```
df["job_title"].value_counts()
```

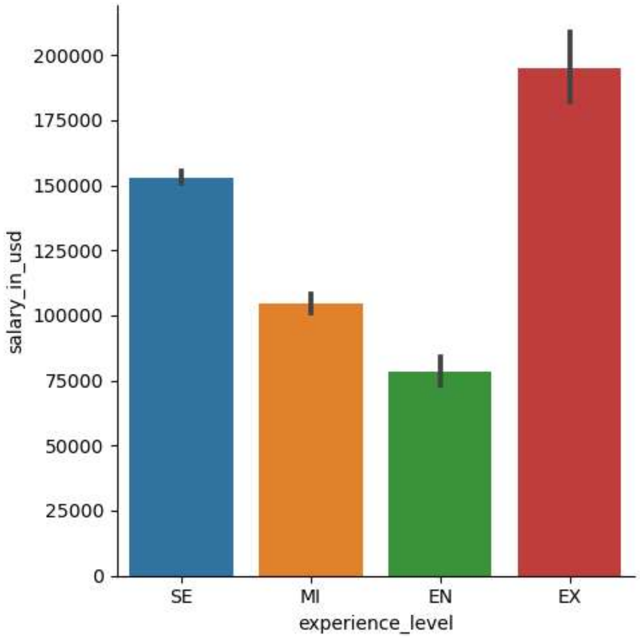
Out[20]:
```
job_title
Data Engineer                        1040
Data Scientist                        840
Data Analyst                          612
Machine Learning Engineer             289
Analytics Engineer                    103
                                     ...
Principal Machine Learning Engineer     1
Azure Data Engineer                     1
Manager Data Management                 1
Marketing Data Engineer                 1
Finance Data Analyst                    1
Name: count, Length: 93, dtype: int64
```

In [21]:
```
job_title_salaries = df.groupby('job_title')['salary_in_usd'].mean().sort_values(ascending=False)

# Create horizontal bar chart
fig, ax = plt.subplots(figsize=(8, 20))
ax.barh(job_title_salaries.index, job_title_salaries.values)
ax.set_title('Average Salaries by Job Title')
ax.set_xlabel('Salary in USD')
ax.set_ylabel('Job Title')
plt.show()
```


Average Salaries by Job Title
Job Title
Power BI Developer
Product Data Scientist
Staff Data Analyst
3D Computer Vision Researcher
Autonomous Vehicle Technician
Compliance Data Analyst
Principal Data Architect
Machine Learning Research Engineer
Insight Analyst
Data DevOps Engineer
AI Programmer
Product Data Analyst
BI Data Engineer
BI Data Analyst
Finance Data Analyst
Software Data Engineer
Business Data Analyst
Marketing Data Engineer
Data Analytics Engineer
Data Management Specialist
Big Data Engineer
ETL Engineer
Lead Data Analyst
Machine Learning Developer
Head of Machine Learning
Computer Vision Software Engineer
Data Strategist
Data Analytics Consultant
Lead Machine Learning Engineer
Data Quality Analyst
Data Science Consultant
Cloud Data Engineer
Data Operations Analyst
Data Science Engineer
Data Analytics Specialist
Data Operations Engineer
Applied Machine Learning Engineer
Azure Data Engineer
Financial Data Analyst
BI Analyst
Staff Data Scientist
Machine Learning Researcher
Data Analyst
Deep Learning Engineer
Applied Machine Learning Scientist
AI Scientist
Lead Data Scientist
Applied Data Scientist
Data Manager
Data Modeler
Data Specialist
Principal Data Analyst
Deep Learning Researcher
Manager Data Management
Big Data Architect
MLOps Engineer
BI Developer
ETL Developer
NLP Engineer
Data Scientist Lead
AI Developer
Lead Data Engineer
Data Scientist
Data Analytics Manager
Data Engineer
Machine Learning Infrastructure Engineer
Computer Vision Engineer
Marketing Data Analyst
Analytics Engineer
Machine Learning Engineer
Cloud Database Engineer
Machine Learning Manager
Data Science Lead
ML Engineer
Head of Data Science
Research Scientist
Data Architect
Research Engineer
Machine Learning Scientist
Business Intelligence Engineer
Data Infrastructure Engineer
Head of Data
Principal Machine Learning Engineer
Applied Scientist
Data Science Manager
Machine Learning Software Engineer
Principal Data Engineer
Director of Data Science
Principal Data Scientist
Data Analytics Lead
Data Lead

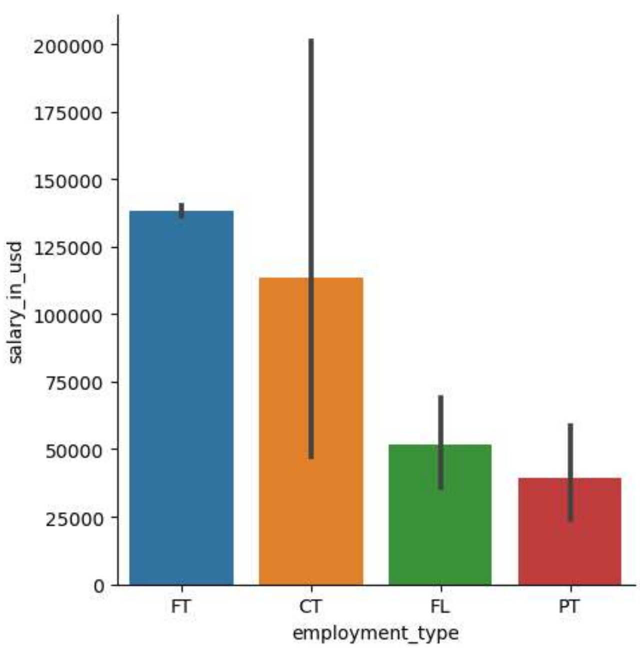


In [22]:
```
sns.catplot(x="experience_level",y="salary_in_usd" ,kind="bar",data=df)
plt.show()
```

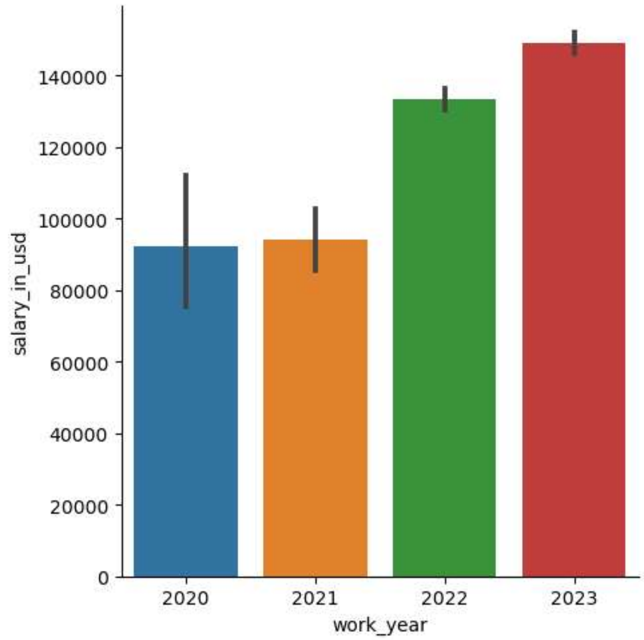


In [ ]:
```
There are 4 categorical values in the column "employment_type", such as:

FT, which is Full-time. PT, which is Part-time. CT, which is Contractual. FL, which is Freelancer.
```

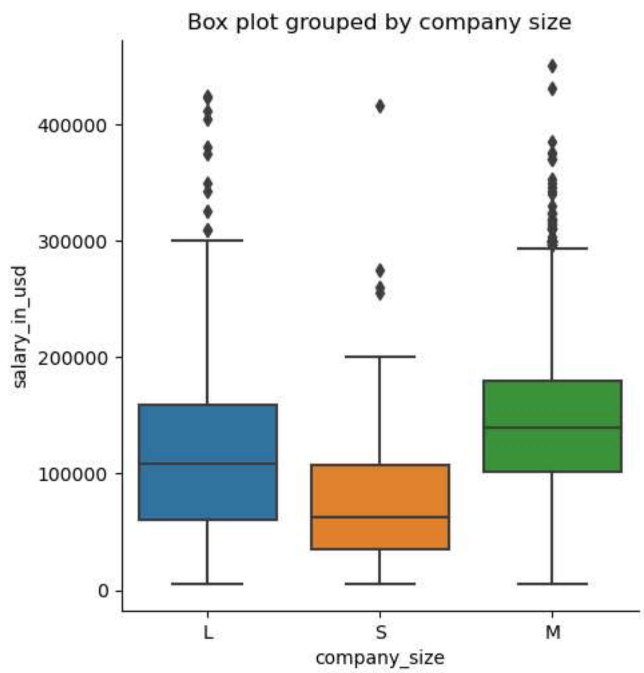
In [23]:
```
sns.catplot(x="employment_type",y="salary_in_usd" ,kind="bar",data=df)
plt.show()
```



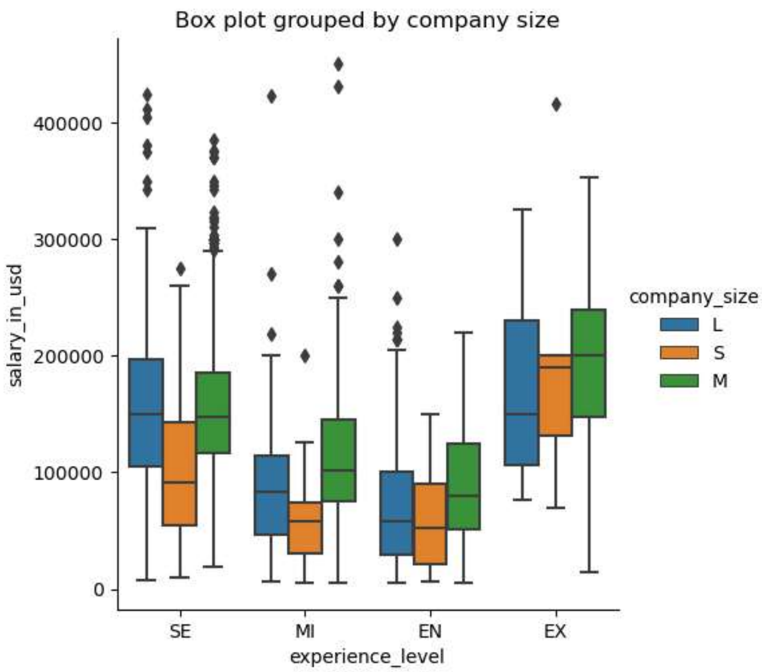
In [24]:
```
sns.catplot(x="work_year",y="salary_in_usd" ,kind="bar",data=df)
plt.show()
```



In [25]:
```
sns.catplot(x="company_size",y="salary_in_usd",kind="box",data=df)
plt.title("Box plot grouped by company size")
plt.show()
```



In [26]:
```
sns.catplot(x="experience_level",y="salary_in_usd",hue="company_size" ,kind="box",data=df)
plt.title("Box plot grouped by company size")
plt.show()
```



In [27]:
```
cat_list=[i for i in df.select_dtypes("object")]
```

In [28]:
```
cat_list
```

Out[28]:
```
['experience_level',
 'employment_type',
 'job_title',
 'salary_currency',
 'employee_residence',
 'company_location',
 'company_size']
```

In [29]:
```
for i in cat_list:
    df[i] = df[i].factorize()[0]
```

In [30]:
```
df
```

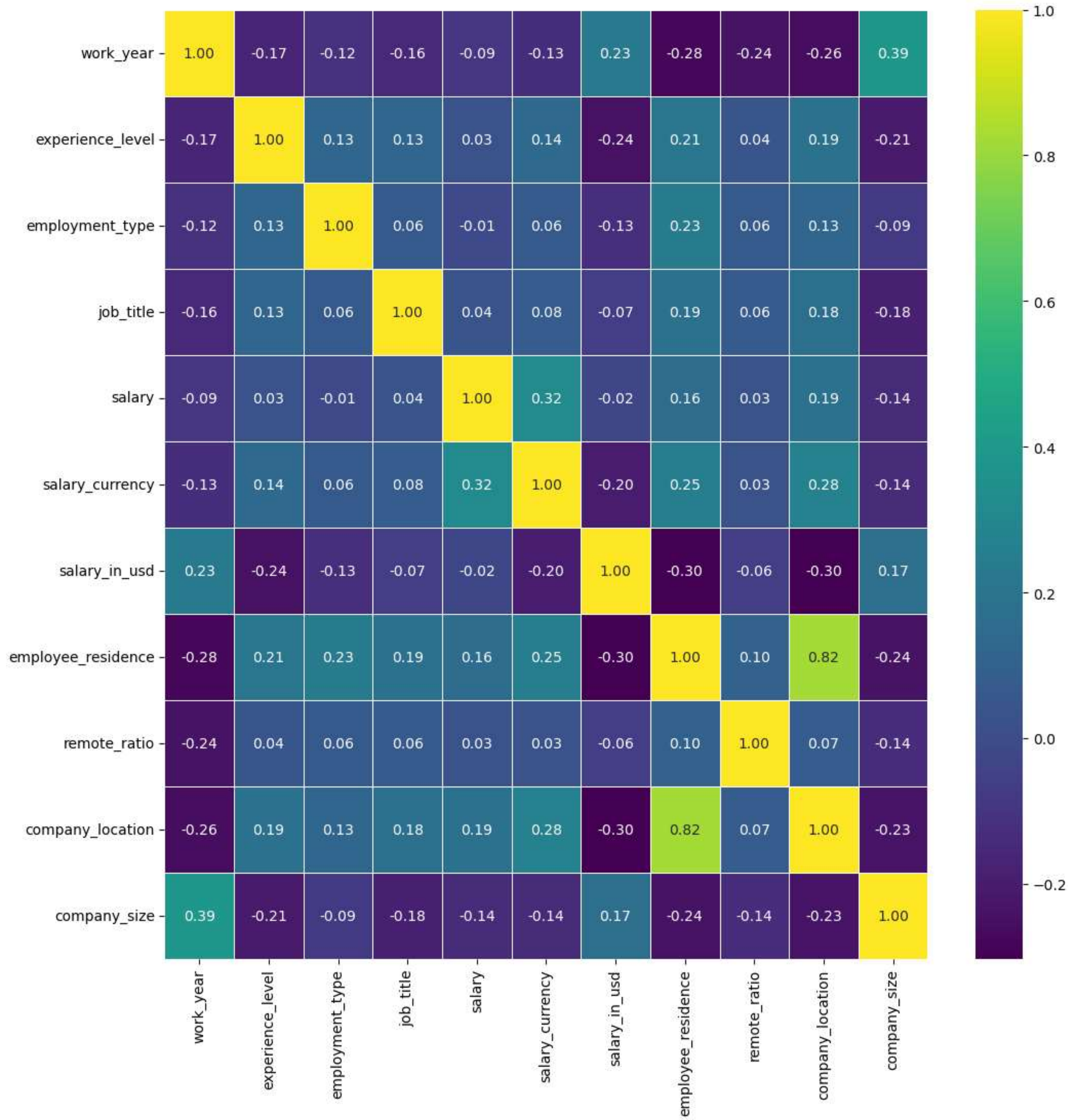
Out[30]:

	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_residence	remote_ratio	company_location	c
0	2023	0	0	0	80000	0	85847	0	100	0	
1	2023	1	1	1	30000	1	30000	1	100	1	
2	2023	1	1	1	25500	1	25500	1	100	1	
3	2023	0	0	2	175000	1	175000	2	100	2	
4	2023	0	0	2	120000	1	120000	2	100	2	
...	...	...	...	...	...	...	...	...	...	...	
3750	2020	0	0	2	412000	1	412000	1	100	1	
3751	2021	1	0	0	151000	1	151000	1	100	1	
3752	2020	2	0	2	105000	1	105000	1	100	1	
3753	2020	2	1	20	100000	1	100000	1	100	1	
3754	2021	0	0	26	7000000	2	94665	6	50	6	

3755 rows × 11 columns

In [ ]:
```
Correlation
```

In [31]:
```
plt.figure(figsize=(12,12))
sns.heatmap(df.corr(),annot=True,linewidths=0.7,cmap="viridis",fmt=".2f")
plt.show()
```



In [32]:
```
X=df.drop(["salary_in_usd"], axis = 1)
Y=df["salary_in_usd"]
```

In [ ]:
```
Splitting the Dataset into Train & Test:
```

In [33]:
```
from sklearn.model_selection import train_test_split
```

In [34]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

In [35]:
```
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(3004, 10)
(751, 10)
(3004,)
(751,)
```

In [ ]:
```
Modeling
```

In [36]:
```
from sklearn.linear_model import Ridge,Lasso,RidgeCV,LassoCV,ElasticNet,ElasticNetCV,LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn import neighbors
from sklearn.svm import SVR
```

In [37]:
```
dt=DecisionTreeRegressor()
```

In [38]:
```
dt.fit(X_train,Y_train)
```

Out[38]:
```
▸ DecisionTreeRegressor
```

In [39]:
```
y_predict = dt.predict(X_test)
```

In [40]:
```
dt.score(X_train,Y_train)
```

Out[40]:
```
1.0
```

In [41]:
```
dt.score(X_test,Y_test)
```

Out[41]:
```
0.9951895431762576
```

In [42]:
```
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

In [43]:
```
print(r2_score(Y_test, y_predict)*100)
print(mean_squared_error(Y_test, y_predict))
print(mean_absolute_error(Y_test, y_predict))
```

```
99.51895431762577
18990672.55259654
714.9147802929427
```

In [44]:
```
knn=KNeighborsRegressor().fit(X_train,Y_train)
ada=AdaBoostRegressor().fit(X_train,Y_train)
svm=SVR().fit(X_train,Y_train)
ridge=Ridge().fit(X_train,Y_train)
lasso=Lasso().fit(X_train,Y_train)
rf=RandomForestRegressor().fit(X_train,Y_train)
gbm=GradientBoostingRegressor().fit(X_train,Y_train)
```

In [45]:
```
models=[ridge,lasso,knn,ada,svm,rf,gbm]
```

In [46]:
```
def ML(Y,models):
    y_pred=models.predict(X_test)
    mse=mean_squared_error(Y_test,y_pred)
    rmse=np.sqrt(mean_squared_error(Y_test,y_pred))
    r2=r2_score(Y_test,y_pred)*100

    return mse,rmse,r2
```

In [47]:
```
for i in models:
    print("\n",i,"\n\nDifferent models success rate :",ML("salary_in_usd",i))
```

```
 Ridge()

Different models success rate : (3464466548.979176, 58859.71923972434, 12.242882894184092)

 Lasso()

Different models success rate : (3464396847.135122, 58859.12713534853, 12.244648485741894)

 KNeighborsRegressor()

Different models success rate : (390614700.0352064, 19763.97480354613, 90.10548392093921)

 AdaBoostRegressor()

Different models success rate : (189138459.0234362, 13752.761868927862, 95.20900385006632)

 SVR()

Different models success rate : (3944880838.892905, 62808.28638717112, 0.07368671254132098)

 RandomForestRegressor()

Different models success rate : (22340867.97563116, 4726.612738064243, 99.4340917600256)

 GradientBoostingRegressor()

Different models success rate : (12715598.669316212, 3565.893810717898, 99.67790588660105)
```

In [ ]:
```
The models can be arranged in the following order of best to worst performance using the evaluation metrics that have been provic
Regressors with gradient boosting, random forest, decision tree, and AdaBoost are arranged in order of preference, followed by ne
```