In [2]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```python
df = pd.read_csv("ds_salaries.csv")
```

In [4]:
```python
df
```

Out[4]:

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

3755 rows × 11 columns

In [5]:
```python
df.head(10)
```

Out[5]:

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

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

In [6]:
```python
df.shape
```

Out[6]:
```
(3755, 11)
```

In [7]:
```python
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 [8]:
```python
df.columns
```

Out[8]:
```
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 [9]:
```python
df.describe()
```

Out[9]:

	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 [10]:
```python
df.dtypes
```

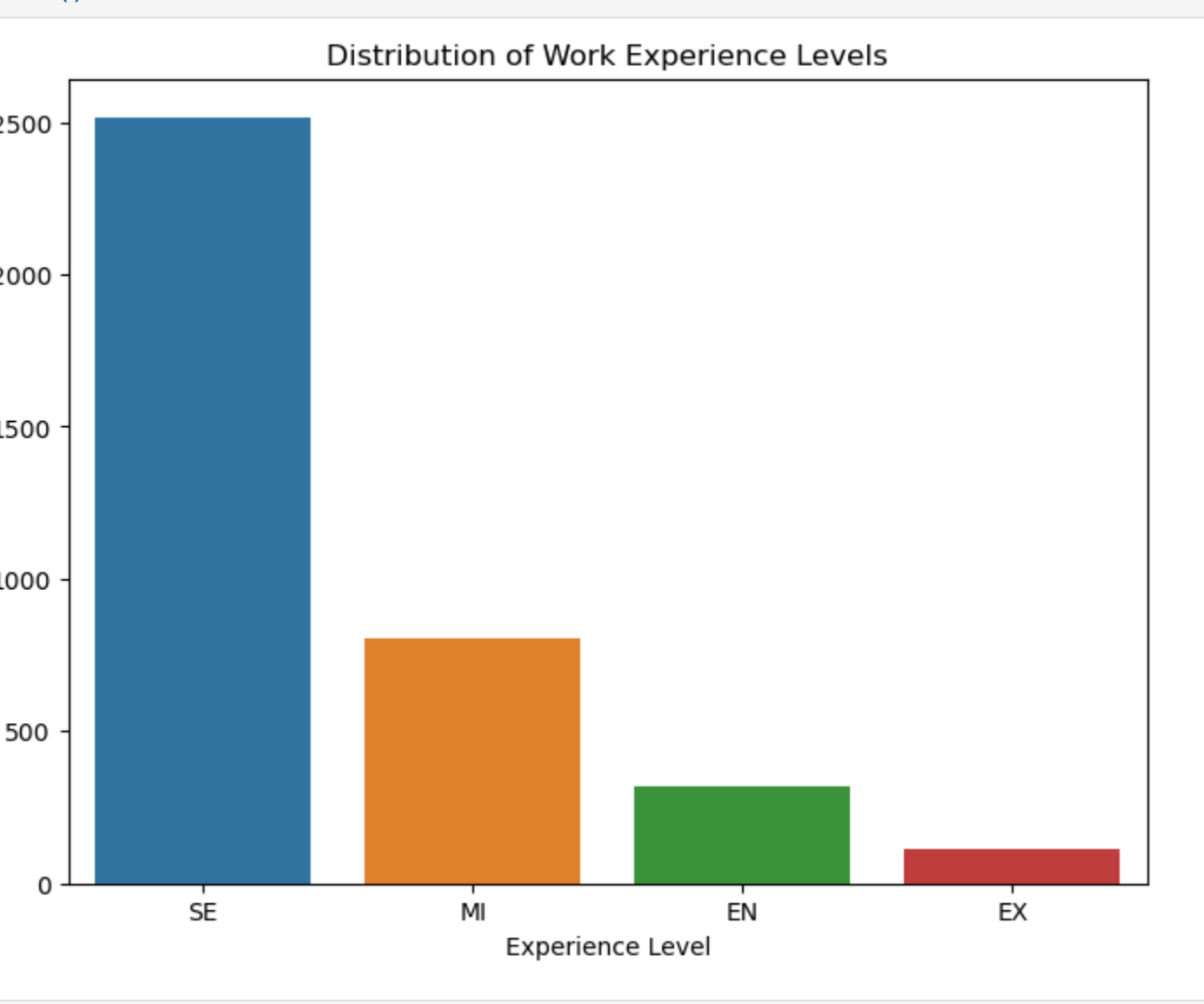
Out[10]:
```
work_year              int64
experience_level      object
employment_type       object
job_title             object
salary                 int64
salary_currency       object
salary_in_usd          int64
employee_residence    object
remote_ratio           int64
company_location      object
company_size          object
dtype: object
```

In [11]:
```python
df.nunique()
```

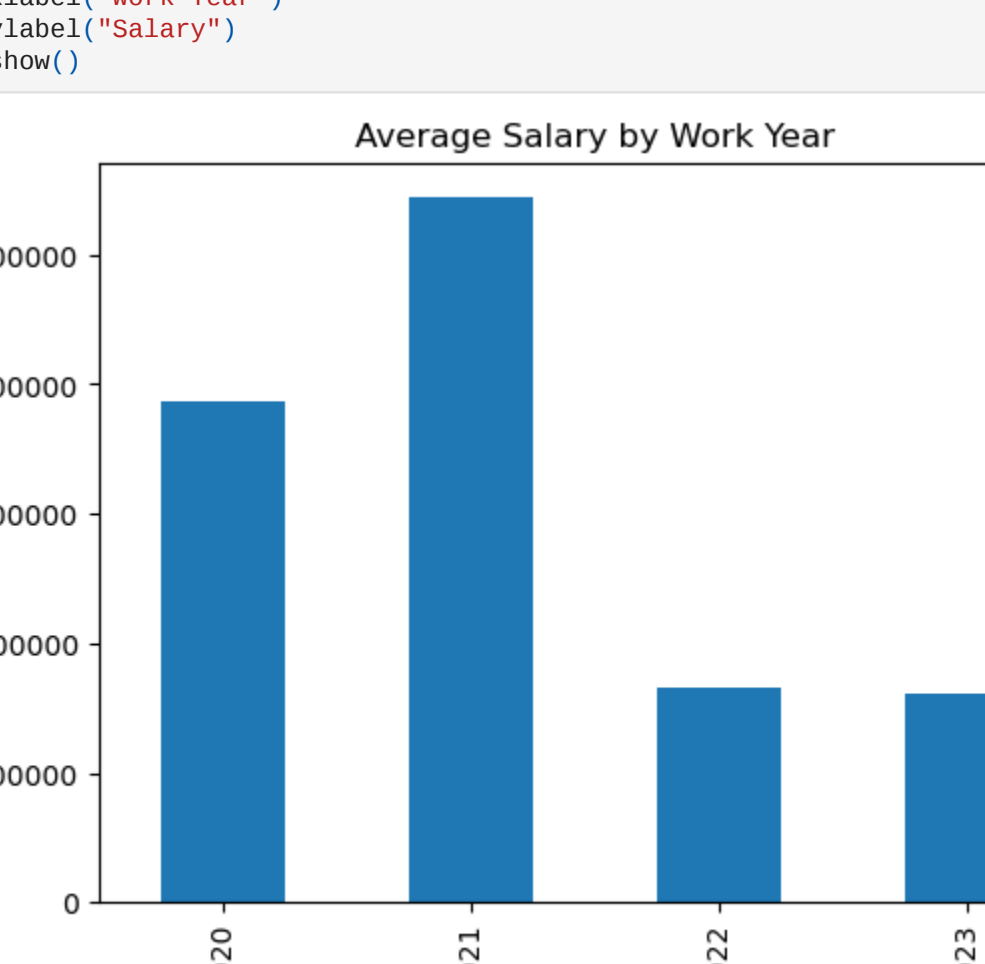
Out[11]:
```
work_year                4
experience_level         4
employment_type          4
job_title               93
salary                 815
salary_currency         20
salary_in_usd         1035
employee_residence      78
remote_ratio             3
company_location        72
company_size             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
```

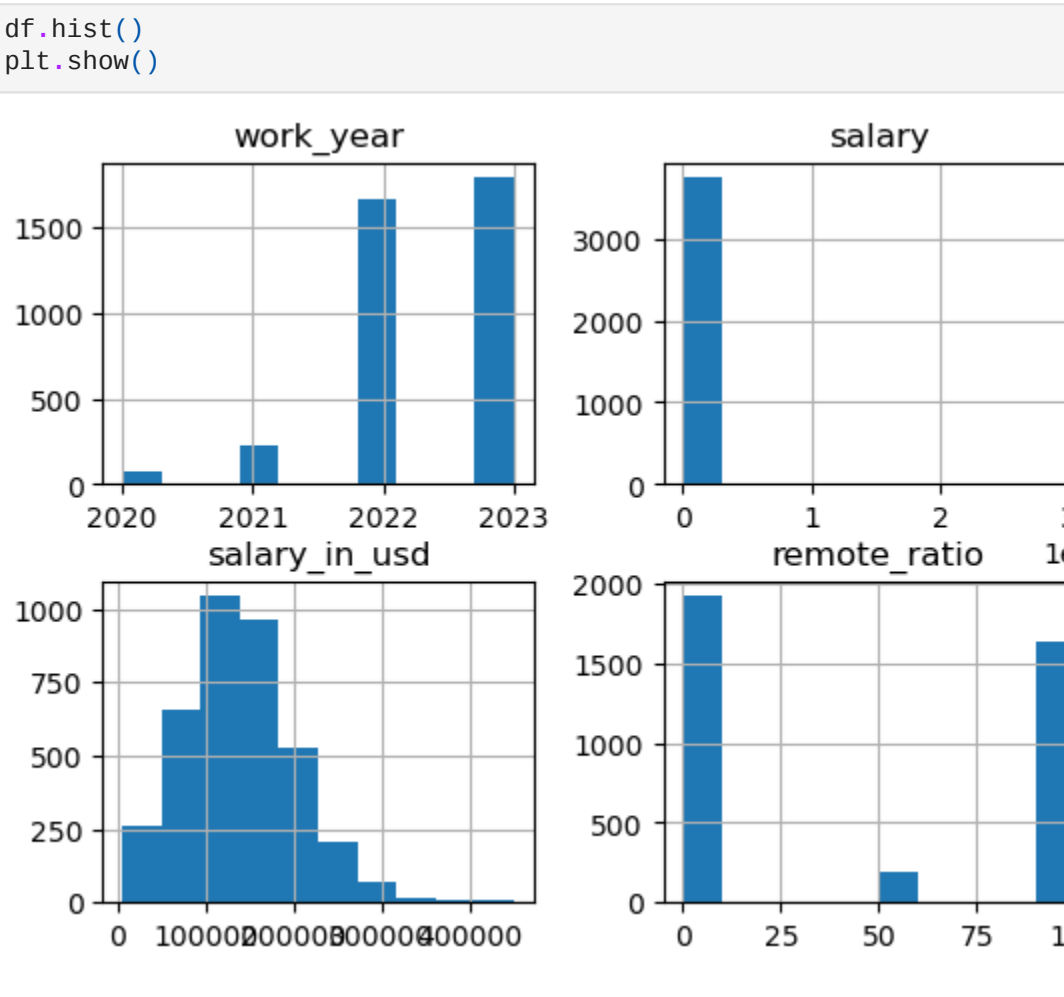
In [14]:
```python
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='experience_level')
plt.xlabel('Experience Level')
plt.ylabel('Count')
plt.title('Distribution of Work Experience Levels')
plt.show()
```



In [17]:
```python
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()
```



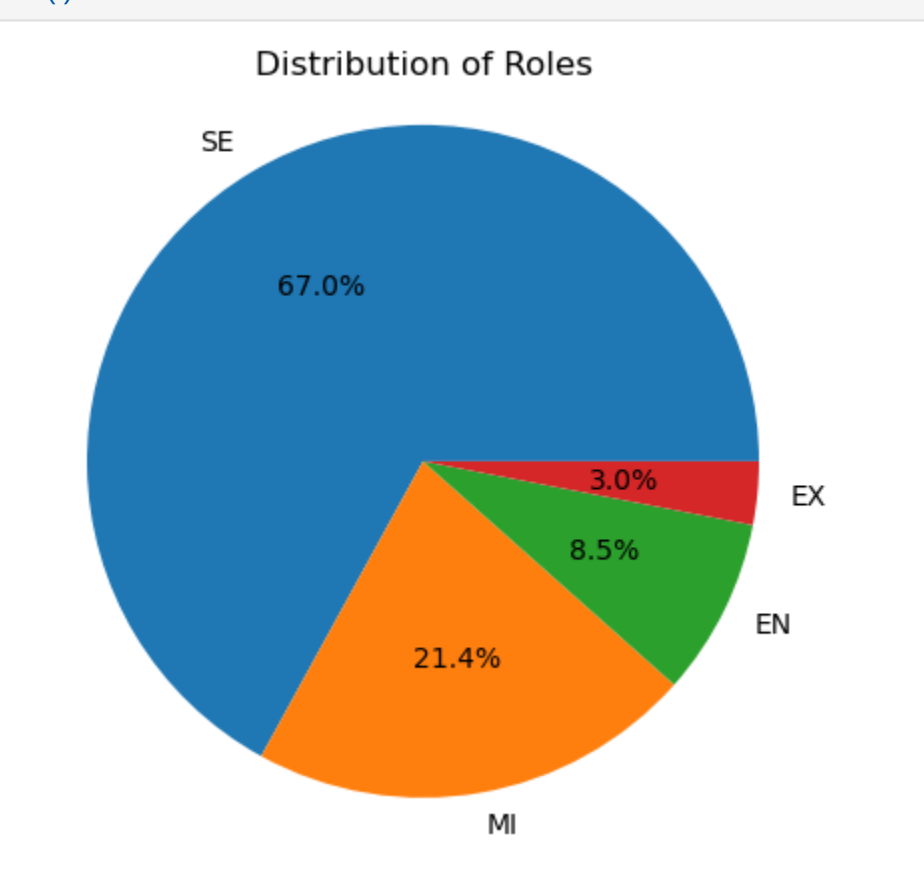
In [18]:
```python
df.hist()
plt.show()
```



In [19]:
```python
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
```

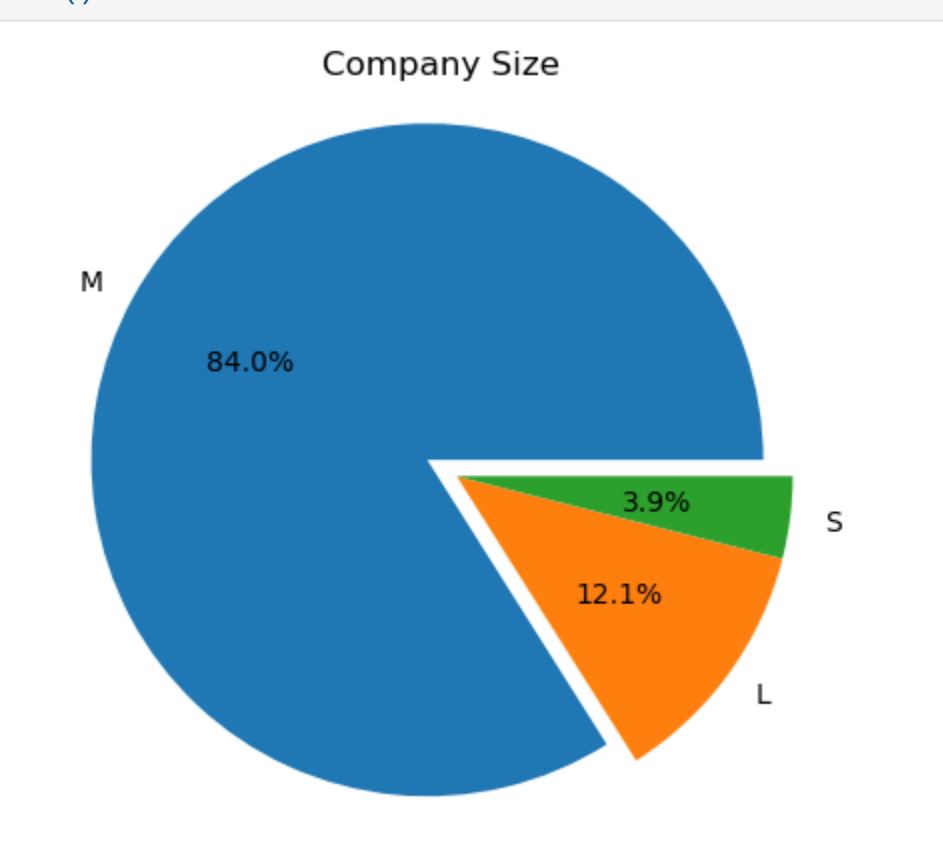
In [20]:
```python
roles = ["SE", "MI", "EN", "EX"]
people = [2516, 805, 320, 114]

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



In [21]:
```python
print(df["employment_type"].value_counts())
```

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

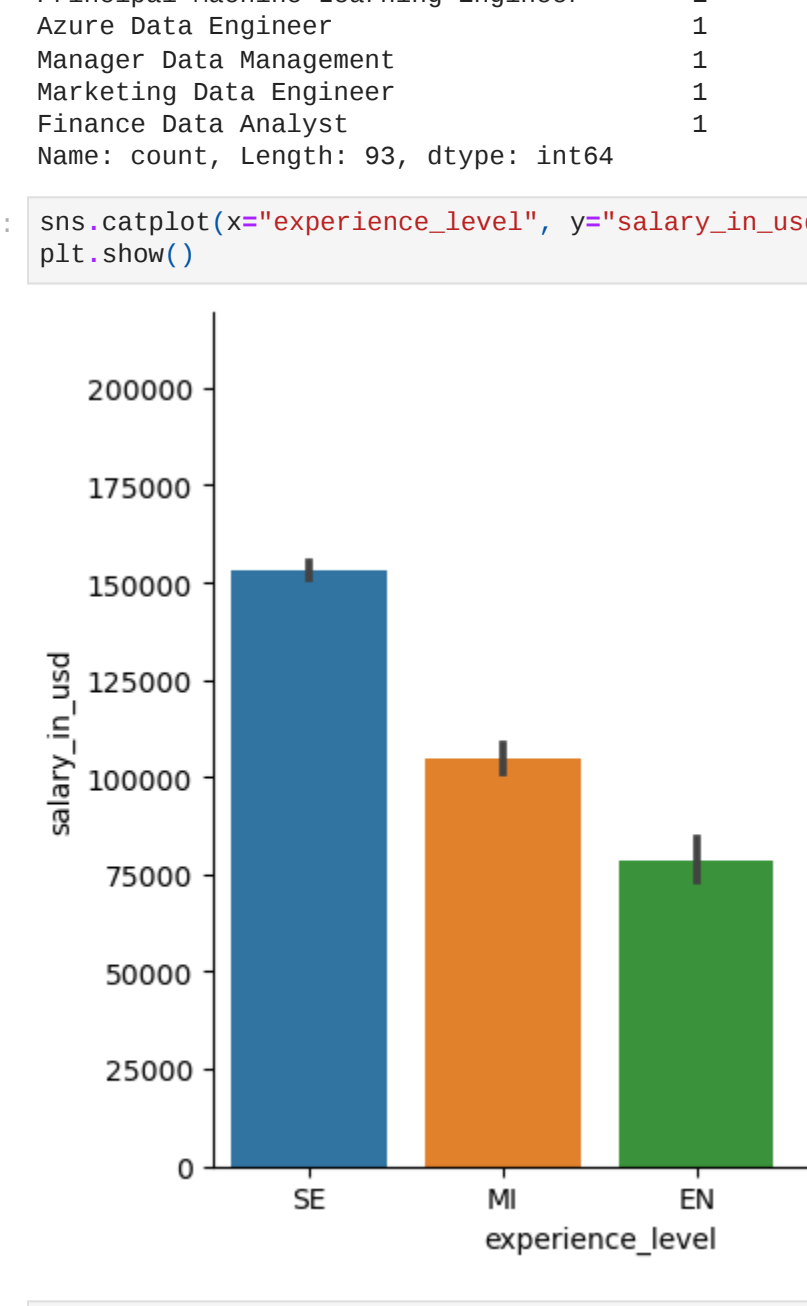
In [22]:
```python
company_numbers = [3153, 454, 148]
company_size = ["M", "L", "S"]
explode = [0.1, 0, 0]

plt.pie(company_numbers, labels=company_size, explode=explode, autopct='%1.1f%%')
plt.axis('equal')
plt.title("Company Size")
plt.show()
```



In [23]:
```python
df["job_title"].value_counts()
```

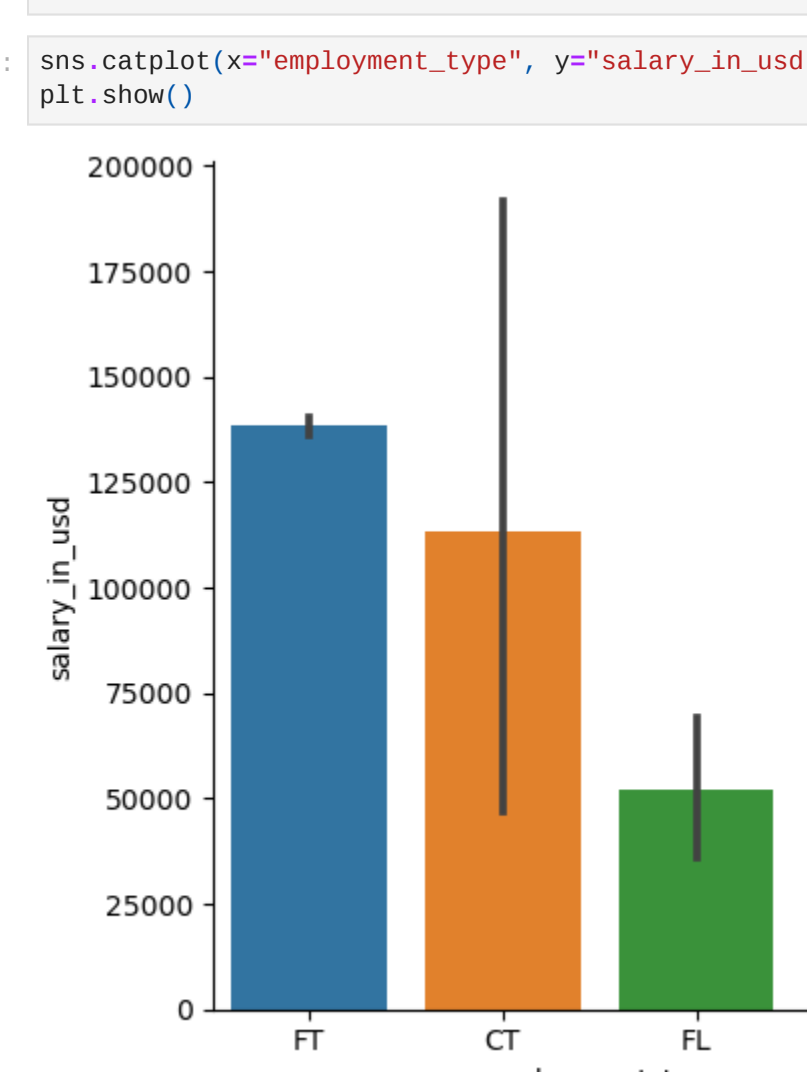
Out[23]:
```
job_title
Data Engineer                          1040
Data Scientist                          840
Data Analyst                            612
Machine Learning Engineer               289
Analytics Engineer                      103
                                       ... 
Principal Machine 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 [24]:
```python
sns.catplot(x="experience_level", y="salary_in_usd", kind="bar", data=df)
plt.show()
```

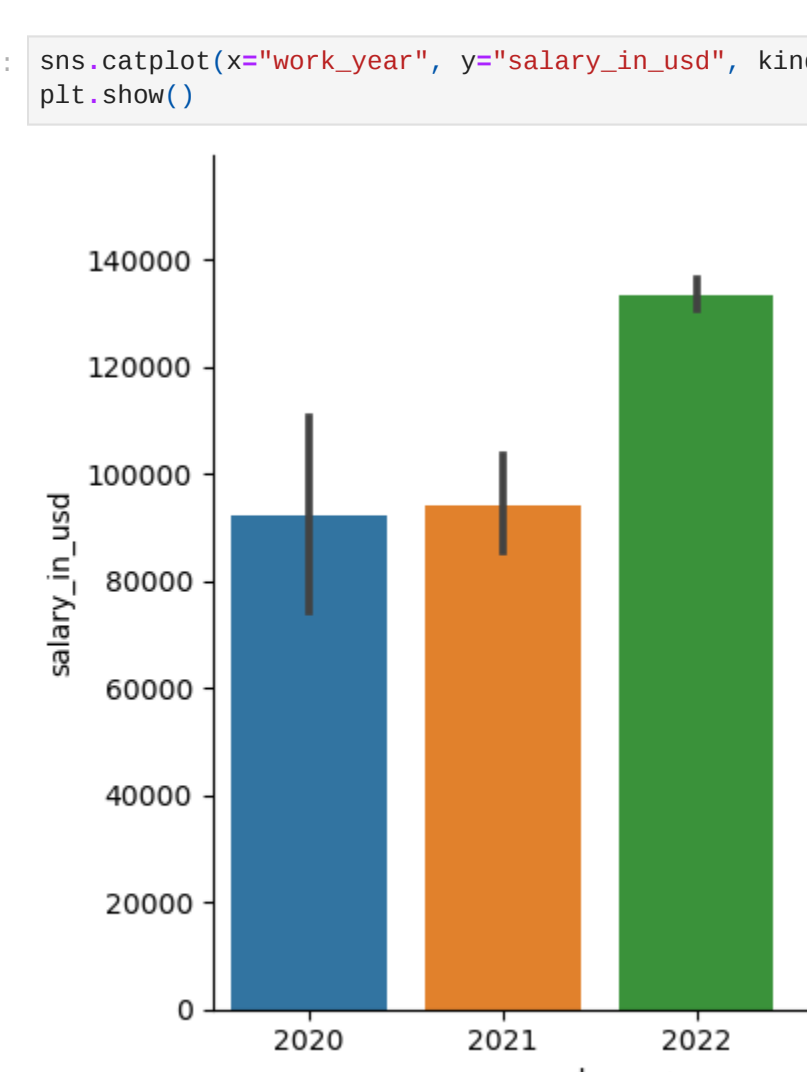


In [ ]:
```
There are 4 categorical values in the column "employment_type", such as:
FT, which is Full-time. PT, which is Pant-time. CT, which is Contractual. FL, which is Fneelancer.
```

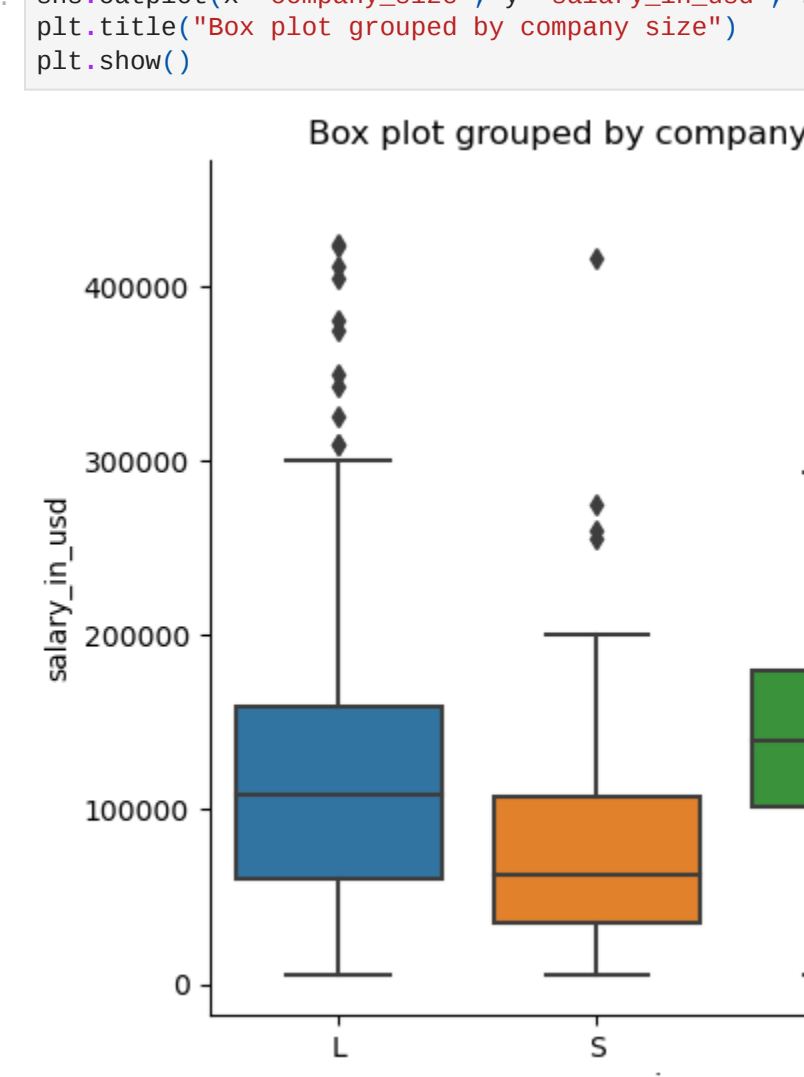
In [26]:
```python
sns.catplot(x="employment_type", y="salary_in_usd", kind="bar", data=df)
plt.show()
```



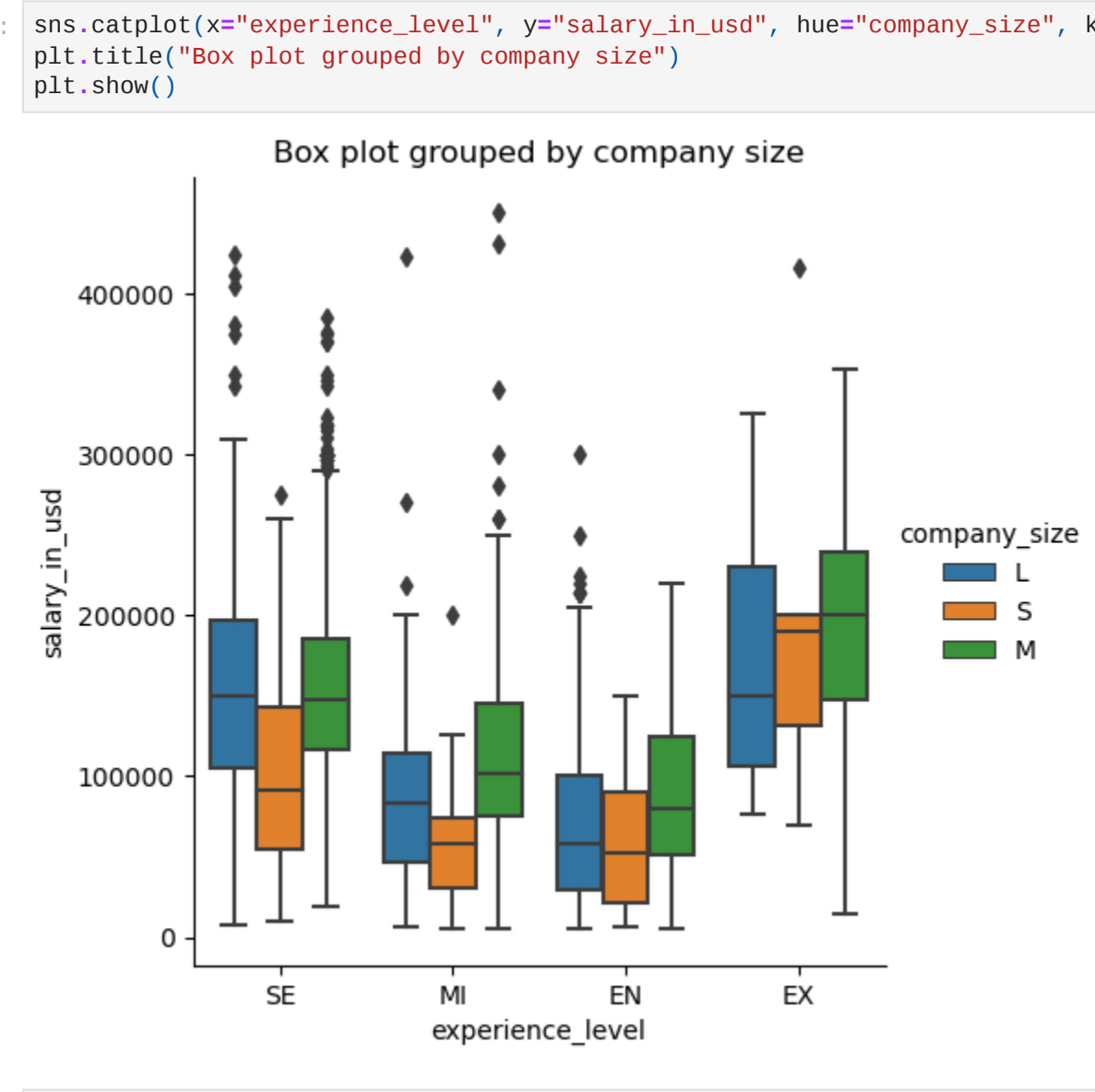
In [27]:
```python
sns.catplot(x="work_year", y="salary_in_usd", kind="bar", data=df)
plt.show()
```



In [28]:
```python
sns.catplot(x="company_size", y="salary_in_usd", kind="box", data=df)
plt.title("Box plot grouped by company size")
plt.show()
```



In [29]:
```python
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 [30]:
```python
cat_list = [i for i in df.select_dtypes("object")]
```

In [31]:
```python
cat_list
```

Out[31]:
```
['experience_level',
 'employment_type',
 'job_title',
 'salary_currency',
 'employee_residence',
 'company_location',
 'company_size']
```

In [33]:
```python
for i in cat_list:
    df[i] = df[i].factorize()[0]
```

In [34]:
```python
df
```

Out[34]:

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

3755 rows × 11 columns

In [ ]:
```
Correlat ton
```

In [35]:
```python
plt.figure(figsize=(12, 12))
sns.heatmap(df.corr(), annot=True, linewidths=0.7, cmap="viridis", fmt=".2f")
plt.show()
```

In [36]:
```python
X = df.drop(["salary_in_usd"], axis=1)
Y = df["salary_in_usd"]
```

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

In [37]:
```python
from sklearn.model_selection import train_test_split
```

In [38]:
```python
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

In [39]:
```python
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("Y_train shape:", Y_train.shape)
print("Y_test shape:", Y_test.shape)
```

```
X_train shape: (3004, 10)
X_test shape: (751, 10)
Y_train shape: (3004,)
Y_test shape: (751,)
```

In [ ]:
```
Modeling
```

In [40]:
```python
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, GradientBoostingRegressor, AdaBoostRegressor
from sklearn import neighbors
from sklearn.svm import SVR
```

In [41]:
```python
dt=DecisionTreeRegressor()
```

In [42]:
```python
dt.fit(X_train,Y_train)
```

Out[42]:
```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

In [43]:
```python
y_predict = dt.predict(X_test)
```

In [44]:
```python
dt.score(X_train,Y_train)
```

Out[44]:
```
1.0
```

In [45]:
```python
dt.score(X_test,Y_test)
```

Out[45]:
```
0.9939366761794373
```

In [46]:
```python
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
```

In [47]:
```python
print(r2_score(Y_test, y_predict) * 100)
print(mean_squared_error(Y_test, y_predict))
print(mean_absolute_error(Y_test, y_predict))
```

```
99.39366761794372
23936728.14780293
801.9800266311585
```

In [48]:
```python
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 [49]:
```python
models=[ridge,lasso,knn,ada,svm,rf,gbm]
```

In [50]:
```python
def ML(Y, model):
    y_pred = model.predict(X_test)
    mse = mean_squared_error(Y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(Y_test, y_pred) * 100
    return mse, rmse, r2
```

In [51]:
```python
for model in models:
    print("\n", model, "\n\nDifferent models success rate:", ML("salary_in_usd", model))
```

```
 Ridge()

Different models success rate: (3464466548.979133, 58859.71923972398, 12.24288289418518)

 Lasso()

Different models success rate: (3464396847.135084, 58859.12713534821, 12.244648485742848)

 KNeighborsRegressor()

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

 AdaBoostRegressor()

Different models success rate: (227090924.79969683, 15069.536316678654, 94.24764401688705)

 SVR()

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

 RandomForestRegressor()

Different models success rate: (21217502.093839817, 4606.245987117906, 99.46254732449627)

 GradientBoostingRegressor()

Different models success rate: (16574508.687009491, 4071.180257248442, 99.58015726829696)
```

In [ ]: