configuración del entorno

completar líneas

In [1]:
```
%config Completer.use_jedi = True
```

Obtener dataset

Instalar kagglehub

In [2]:
```
conda install kagglehub
```

```
Retrieving notices: done
Note: you may need to restart the kernel to use updated packages.

Channels:
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): ...working... done
Solving environment: ...working... done

# All requested packages already installed.
```

importa kaggle, pandas y numpy , y descargar data

In [10]:
```
import kagglehub  #descargar dataset
import pandas as pd #procesos de tabla
import numpy as np  #procesos de vectores y matemáticas

#visualizacion
import plotly.express as px
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm
```

In [4]:
```
# Download latest version
path = kagglehub.dataset_download("ruchi798/data-science-job-salaries")
```

```
print("Path to dataset files:", path)
```

```
Warning: Looks like you're using an outdated `kagglehub` version, please consider updating (latest version: 0.3.10)
Path to dataset files: C:\Users\darly\.cache\kagglehub\datasets\ruchi798\data-science-job-salaries\versions\1
```

crear un data frame, una tabla como ejemplo

In [11]:
```
data= pd.DataFrame({
    "nombres": ["ana", "juana", "sara"],
    "edad": [12,23,34]
})
data
```

Out[11]:

	nombres	edad
0	ana	12
1	juana	23
2	sara	34

In [12]:
```
data2= pd.DataFrame({
    "nombres": ["ana", "juana", "sara"],
    "salario": [120,230,340]
})
data2
```

Out[12]:

	nombres	salario
0	ana	120
1	juana	230
2	sara	340

unir data_frame

In [7]:
```
new_df= data.merge(data2)
```

In [8]:
```
new_df
```

Out[8]:

	nombres	edad	salario
0	ana	12	120
1	juana	23	230
2	sara	34	340

leer un archivo csv, ya descargado, e imprimir la cabeza (primero 5 elementos)

In [9]:
```
df = pd.read_csv("C:/Users/darly/.cache/kagglehub/datasets/ruchi798/data-science-job-salaries/versions/1/ds_salaries.
```

In [13]:
```
df = pd.read_csv(r"C:\Users\darly\.cache\kagglehub\datasets\ruchi798\data-science-job-salaries\versions\1\ds_salaries
```

exploración, filtro y limpieza de la data

mostrar las primeras 5 filas

In [14]:
```
df.head()
```

Out[14]:

	Unnamed: 0	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_resider
0	0	2020	MI	FT	Data Scientist	70000	EUR	79833	
1	1	2020	SE	FT	Machine Learning Scientist	260000	USD	260000	
2	2	2020	SE	FT	Big Data Engineer	85000	GBP	109024	
3	3	2020	MI	FT	Product Data Analyst	20000	USD	20000	
4	4	2020	SE	FT	Machine Learning Engineer	150000	USD	150000	

mostrar las últimas 5 lineas

In [15]:
```
df.tail()
```

Out[15]:

	Unnamed: 0	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_resid
602	602	2022	SE	FT	Data Engineer	154000	USD	154000	
603	603	2022	SE	FT	Data Engineer	126000	USD	126000	
604	604	2022	SE	FT	Data Analyst	129000	USD	129000	
605	605	2022	SE	FT	Data Analyst	150000	USD	150000	
606	606	2022	MI	FT	AI Scientist	200000	USD	200000	

para describir la data, muestra un resumen del dataset solo en las variables numericas

In [16]:
```
df.describe()
```

Out[16]:

	Unnamed: 0	work_year	salary	salary_in_usd	remote_ratio
count	607.000000	607.000000	6.070000e+02	607.000000	607.00000
mean	303.000000	2021.405272	3.240001e+05	112297.869852	70.92257
std	175.370085	0.692133	1.544357e+06	70957.259411	40.70913
min	0.000000	2020.000000	4.000000e+03	2859.000000	0.00000
25%	151.500000	2021.000000	7.000000e+04	62726.000000	50.00000
50%	303.000000	2022.000000	1.150000e+05	101570.000000	100.00000
75%	454.500000	2022.000000	1.650000e+05	150000.000000	100.00000
max	606.000000	2022.000000	3.040000e+07	600000.000000	100.00000

muestra una lista con todas las columnas que tiene el data frame

In [17]:
```
df.columns
```

Out[17]:
```
Index(['Unnamed: 0', 'work_year', 'experience_level', 'employment_type',
       'job_title', 'salary', 'salary_currency', 'salary_in_usd',
       'employee_residence', 'remote_ratio', 'company_location',
       'company_size'],
      dtype='object')
```

esto sirve para hacer consultas especificas del dataframe

In [18]:
```
df[df.salary_in_usd > 250000]
```

Out[18]:

	Unnamed: 0	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_resi
1	1	2020	SE	FT	Machine Learning Scientist	260000	USD	260000	
25	25	2020	EX	FT	Director of Data Science	325000	USD	325000	
33	33	2020	MI	FT	Research Scientist	450000	USD	450000	
63	63	2020	SE	FT	Data Scientist	412000	USD	412000	
78	78	2021	MI	CT	ML Engineer	270000	USD	270000	
93	93	2021	SE	FT	Lead Data Engineer	276000	USD	276000	
97	97	2021	MI	FT	Financial Data Analyst	450000	USD	450000	
157	157	2021	MI	FT	Applied Machine Learning Scientist	423000	USD	423000	
225	225	2021	EX	CT	Principal Data Scientist	416000	USD	416000	
231	231	2021	SE	FT	ML Engineer	256000	USD	256000	
252	252	2021	EX	FT	Principal Data Engineer	600000	USD	600000	

	Unnamed: 0	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_resi
416	416	2022	SE	FT	Data Scientist	260000	USD	260000	
482	482	2022	EX	FT	Data Engineer	324000	USD	324000	
519	519	2022	SE	FT	Applied Data Scientist	380000	USD	380000	
523	523	2022	SE	FT	Data Analytics Lead	405000	USD	405000	
534	534	2022	SE	FT	Data Architect	266400	USD	266400	

In [19]:
```
df[df.salary_in_usd > 250000].describe()
```

Out[19]:

	Unnamed: 0	work_year	salary	salary_in_usd	remote_ratio
count	16.00000	16.000000	16.000000	16.000000	16.000000
mean	233.06250	2021.062500	360837.500000	360837.500000	78.125000
std	197.70364	0.771902	97733.221066	97733.221066	40.697051
min	1.00000	2020.000000	256000.000000	256000.000000	0.000000
25%	74.25000	2020.750000	269100.000000	269100.000000	87.500000
50%	191.00000	2021.000000	352500.000000	352500.000000	100.000000
75%	432.50000	2022.000000	417750.000000	417750.000000	100.000000
max	534.00000	2022.000000	600000.000000	600000.000000	100.000000

realizar consulta para datos cualitativos

In [20]:
```
df.job_title
```

Out[20]:
```
0                  Data Scientist
1      Machine Learning Scientist
2                Big Data Engineer
3             Product Data Analyst
4       Machine Learning Engineer
                   ...
602                 Data Engineer
603                 Data Engineer
604                  Data Analyst
605                  Data Analyst
606                  AI Scientist
Name: job_title, Length: 607, dtype: object
```

In [21]:
```
df.query("job_title == 'Data Scientist'") #RECUERDE QUE LA CONSULTA QUERY DEBE SER DENTRO DE UNA CADENA
```

Out[21]:

	Unnamed: 0	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_re
0	0	2020	MI	FT	Data Scientist	70000	EUR	79833	
7	7	2020	MI	FT	Data Scientist	11000000	HUF	35735	
10	10	2020	EN	FT	Data Scientist	45000	EUR	51321	
11	11	2020	MI	FT	Data Scientist	3000000	INR	40481	
12	12	2020	EN	FT	Data Scientist	35000	EUR	39916	
...	...	...	...	...	...	...	...	...	
592	592	2022	SE	FT	Data Scientist	230000	USD	230000	
593	593	2022	SE	FT	Data Scientist	150000	USD	150000	
596	596	2022	SE	FT	Data Scientist	210000	USD	210000	
598	598	2022	MI	FT	Data Scientist	160000	USD	160000	
599	599	2022	MI	FT	Data Scientist	130000	USD	130000	

143 rows × 12 columns

las filas determinadas

In [22]:
```
df.iloc[20:40]
```

Out[22]:

	Unnamed: 0	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_res
20	20	2020	MI	FT	Machine Learning Engineer	299000	CNY	43331	
21	21	2020	MI	FT	Product Data Analyst	450000	INR	6072	
22	22	2020	SE	FT	Data Engineer	42000	EUR	47899	
23	23	2020	MI	FT	BI Data Analyst	98000	USD	98000	
24	24	2020	MI	FT	Lead Data Scientist	115000	USD	115000	
25	25	2020	EX	FT	Director of Data Science	325000	USD	325000	
26	26	2020	EN	FT	Research Scientist	42000	USD	42000	
27	27	2020	SE	FT	Data Engineer	720000	MXN	33511	
28	28	2020	EN	CT	Business Data Analyst	100000	USD	100000	
29	29	2020	SE	FT	Machine Learning Manager	157000	CAD	117104	
30	30	2020	MI	FT	Data Engineering Manager	51999	EUR	59303	
31	31	2020	EN	FT	Big Data Engineer	70000	USD	70000	

	Unnamed: 0	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_res
32	32	2020	SE	FT	Data Scientist	60000	EUR	68428	
33	33	2020	MI	FT	Research Scientist	450000	USD	450000	
34	34	2020	MI	FT	Data Analyst	41000	EUR	46759	
35	35	2020	MI	FT	Data Engineer	65000	EUR	74130	
36	36	2020	MI	FT	Data Science Consultant	103000	USD	103000	
37	37	2020	EN	FT	Machine Learning Engineer	250000	USD	250000	
38	38	2020	EN	FT	Data Analyst	10000	USD	10000	
39	39	2020	EN	FT	Machine Learning Engineer	138000	USD	138000	

columnas específicas de una dataframe

In [23]:
```
df[["job_title", "salary"]]
```

Out[23]:

	job_title	salary
0	Data Scientist	70000
1	Machine Learning Scientist	260000
2	Big Data Engineer	85000
3	Product Data Analyst	20000
4	Machine Learning Engineer	150000
...	...	...
602	Data Engineer	154000
603	Data Engineer	126000
604	Data Analyst	129000
605	Data Analyst	150000
606	AI Scientist	200000

607 rows × 2 columns

otra forma es con la estructura iloc, pero no dando nombres sino posiciones 8recordar que la primera posicion es filas las demas columnas)

In [24]:
```
df.iloc[:, [2,4,5]]
```

Out[24]:

	experience_level	job_title	salary
0	MI	Data Scientist	70000
1	SE	Machine Learning Scientist	260000
2	SE	Big Data Engineer	85000
3	MI	Product Data Analyst	20000
4	SE	Machine Learning Engineer	150000
...	...	...	...
602	SE	Data Engineer	154000
603	SE	Data Engineer	126000
604	SE	Data Analyst	129000
605	SE	Data Analyst	150000
606	MI	AI Scientist	200000

607 rows × 3 columns

columnas determinadas y filas determinadas (estas ultimas son las primeras)

In [25]:
```
df.iloc[10:40, [2,4,5]]
```

Out[25]:

	experience_level	job_title	salary
10	EN	Data Scientist	45000
11	MI	Data Scientist	3000000
12	EN	Data Scientist	35000
13	MI	Lead Data Analyst	87000
14	MI	Data Analyst	85000
15	MI	Data Analyst	8000
16	EN	Data Engineer	4450000
17	SE	Big Data Engineer	100000
18	EN	Data Science Consultant	423000
19	MI	Lead Data Engineer	56000
20	MI	Machine Learning Engineer	299000
21	MI	Product Data Analyst	450000
22	SE	Data Engineer	42000
23	MI	BI Data Analyst	98000
24	MI	Lead Data Scientist	115000
25	EX	Director of Data Science	325000
26	EN	Research Scientist	42000
27	SE	Data Engineer	720000
28	EN	Business Data Analyst	100000
29	SE	Machine Learning Manager	157000
30	MI	Data Engineering Manager	51999
31	EN	Big Data Engineer	70000

	experience_level	job_title	salary
32	SE	Data Scientist	60000
33	MI	Research Scientist	450000
34	MI	Data Analyst	41000
35	MI	Data Engineer	65000
36	MI	Data Science Consultant	103000
37	EN	Machine Learning Engineer	250000
38	EN	Data Analyst	10000
39	EN	Machine Learning Engineer	138000

las columnas con nombres y no por posicion, desde una a otra

In [26]:
```
df.loc[:,"experience_level": "job_title"]
```

Out[26]:

	experience_level	employment_type	job_title
0	MI	FT	Data Scientist
1	SE	FT	Machine Learning Scientist
2	SE	FT	Big Data Engineer
3	MI	FT	Product Data Analyst
4	SE	FT	Machine Learning Engineer
...	...	...	...
602	SE	FT	Data Engineer
603	SE	FT	Data Engineer
604	SE	FT	Data Analyst
605	SE	FT	Data Analyst
606	MI	FT	AI Scientist

607 rows × 3 columns

otra forma de consultar, parecido al query

In [27]:
```
df.loc[df["experience_level"]== "MI"]
```

Out[27]:

	Unnamed: 0	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_re
0	0	2020	MI	FT	Data Scientist	70000	EUR	79833	
3	3	2020	MI	FT	Product Data Analyst	20000	USD	20000	
7	7	2020	MI	FT	Data Scientist	11000000	HUF	35735	
8	8	2020	MI	FT	Business Data Analyst	135000	USD	135000	
11	11	2020	MI	FT	Data Scientist	3000000	INR	40481	
...	...	...	...	...	...	...	...	...	
567	567	2022	MI	FT	Data Analyst	50000	GBP	65438	
586	586	2022	MI	FT	Data Analyst	35000	GBP	45807	
598	598	2022	MI	FT	Data Scientist	160000	USD	160000	
599	599	2022	MI	FT	Data Scientist	130000	USD	130000	
606	606	2022	MI	FT	AI Scientist	200000	USD	200000	

213 rows × 12 columns

In [28]:
```
df.loc[df["experience_level"]== "MI", ["job_title",     "salary"]]
```

Out[28]:

	job_title	salary
0	Data Scientist	70000
3	Product Data Analyst	20000
7	Data Scientist	11000000
8	Business Data Analyst	135000
11	Data Scientist	3000000
...	...	...
567	Data Analyst	50000
586	Data Analyst	35000
598	Data Scientist	160000
599	Data Scientist	130000
606	AI Scientist	200000

213 rows × 2 columns

In [29]:
```
df.loc[df["experience_level"]== "MI", ["job_title",     "salary"]].sort_values("salary", ascending=True)
```

Out[29]:

	job_title	salary
185	Data Engineer	4000
15	Data Analyst	8000
184	Machine Learning Scientist	12000
192	Big Data Engineer	18000
208	Data Engineer	20000
...	...	...
136	ML Engineer	7000000
137	ML Engineer	8500000
7	Data Scientist	11000000
102	BI Data Analyst	11000000
177	Data Scientist	30400000

213 rows × 2 columns

cambiar el nombre de una columna

In [30]:
```
df.rename(columns= {"salary": "salario"})
```

Out[30]:

	Unnamed: 0	work_year	experience_level	employment_type	job_title	salario	salary_currency	salary_in_usd	employee_resid
0	0	2020	MI	FT	Data Scientist	70000	EUR	79833	
1	1	2020	SE	FT	Machine Learning Scientist	260000	USD	260000	
2	2	2020	SE	FT	Big Data Engineer	85000	GBP	109024	
3	3	2020	MI	FT	Product Data Analyst	20000	USD	20000	
4	4	2020	SE	FT	Machine Learning Engineer	150000	USD	150000	
...	...	...	...	...	...	...	...	...	
602	602	2022	SE	FT	Data Engineer	154000	USD	154000	
603	603	2022	SE	FT	Data Engineer	126000	USD	126000	
604	604	2022	SE	FT	Data Analyst	129000	USD	129000	
605	605	2022	SE	FT	Data Analyst	150000	USD	150000	
606	606	2022	MI	FT	AI Scientist	200000	USD	200000	

607 rows × 12 columns

borrar columnas

In [31]:
```
df.drop(columns={"salary"})
```

Out[31]:

	Unnamed: 0	work_year	experience_level	employment_type	job_title	salary_currency	salary_in_usd	employee_residence	re
0	0	2020	MI	FT	Data Scientist	EUR	79833	DE	
1	1	2020	SE	FT	Machine Learning Scientist	USD	260000	JP	
2	2	2020	SE	FT	Big Data Engineer	GBP	109024	GB	
3	3	2020	MI	FT	Product Data Analyst	USD	20000	HN	
4	4	2020	SE	FT	Machine Learning Engineer	USD	150000	US	
...	...	...	...	...	...	...	...	...	
602	602	2022	SE	FT	Data Engineer	USD	154000	US	
603	603	2022	SE	FT	Data Engineer	USD	126000	US	
604	604	2022	SE	FT	Data Analyst	USD	129000	US	
605	605	2022	SE	FT	Data Analyst	USD	150000	US	
606	606	2022	MI	FT	AI Scientist	USD	200000	IN	

607 rows × 11 columns

agregar una nueva columna o modificarla

In [32]:
```
df["salario en pesos"] = df.salary * 4500
df
```

Out[32]:

	Unnamed: 0	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_resid
0	0	2020	MI	FT	Data Scientist	70000	EUR	79833	
1	1	2020	SE	FT	Machine Learning Scientist	260000	USD	260000	
2	2	2020	SE	FT	Big Data Engineer	85000	GBP	109024	
3	3	2020	MI	FT	Product Data Analyst	20000	USD	20000	
4	4	2020	SE	FT	Machine Learning Engineer	150000	USD	150000	
...	...	...	...	...	...	...	...	...	
602	602	2022	SE	FT	Data Engineer	154000	USD	154000	
603	603	2022	SE	FT	Data Engineer	126000	USD	126000	
604	604	2022	SE	FT	Data Analyst	129000	USD	129000	
605	605	2022	SE	FT	Data Analyst	150000	USD	150000	
606	606	2022	MI	FT	AI Scientist	200000	USD	200000	

607 rows × 13 columns

obtener muestras aleatorias (usos testing)

In [33]:
```
df.sample(frac=0.5) #fragmento deel 50 por ciento de los datos
```

Out[33]:

	Unnamed: 0	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_res
282	282	2021	MI	PT	Data Engineer	59000	EUR	69741	
226	226	2021	SE	FT	Data Scientist	110000	CAD	87738	
603	603	2022	SE	FT	Data Engineer	126000	USD	126000	
216	216	2021	EN	PT	Computer Vision Engineer	180000	DKK	28609	
241	241	2021	MI	FT	Data Analyst	80000	USD	80000	
...	...	...	...	...	...	...	...	...	
479	479	2022	MI	FT	Data Scientist	120000	USD	120000	
403	403	2022	SE	FT	Data Analyst	81666	USD	81666	
531	531	2022	MI	FT	Data Analyst	75000	USD	75000	
431	431	2022	MI	FT	Data Analyst	30000	EUR	32974	
219	219	2021	SE	FT	Data Analytics Manager	140000	USD	140000	

304 rows × 13 columns

In [34]:
```
df.sample(n=100) #numero determinado de muestras
```

Out[34]:

	Unnamed: 0	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_res
469	469	2022	SE	FT	Data Scientist	140000	USD	140000	
493	493	2022	SE	FT	Machine Learning Developer	100000	CAD	78791	
56	56	2020	MI	FT	Data Scientist	34000	EUR	38776	
207	207	2021	SE	FT	Data Engineer	165000	USD	165000	
79	79	2021	EN	FT	Data Analyst	80000	USD	80000	
...	...	...	...	...	...	...	...	...	
317	317	2022	SE	FT	Data Scientist	120160	USD	120160	
15	15	2020	MI	FT	Data Analyst	8000	USD	8000	
345	345	2022	SE	FT	Data Engineer	156600	USD	156600	
392	392	2022	SE	FT	Data Analyst	112900	USD	112900	
76	76	2021	MI	FT	BI Data Analyst	100000	USD	100000	

100 rows × 13 columns

agrupar datos determinados y bajo una medida

In [35]:
```
df.groupby("job_title").mean(numeric_only=True)
```

Out[35]:

job_title	Unnamed: 0	work_year	salary	salary_in_usd	remote_ratio	salario en pesos
3D Computer Vision Researcher	77.000000	2021.000000	4.000000e+05	5409.000000	50.000000	1.800000e+09
AI Scientist	254.142857	2021.142857	2.905714e+05	66135.571429	78.571429	1.307571e+09
Analytics Engineer	458.250000	2022.000000	1.750000e+05	175000.000000	50.000000	7.875000e+08
Applied Data Scientist	351.600000	2021.600000	1.724000e+05	175655.000000	70.000000	7.758000e+08
Applied Machine Learning Scientist	321.000000	2021.500000	1.413500e+05	142068.750000	87.500000	6.360750e+08
BI Data Analyst	106.333333	2020.833333	1.902045e+06	74755.166667	66.666667	8.559204e+09
Big Data Architect	255.000000	2021.000000	1.250000e+05	99703.000000	50.000000	5.625000e+08
Big Data Engineer	123.125000	2020.625000	4.550000e+05	51974.000000	50.000000	2.047500e+09
Business Data Analyst	256.800000	2021.000000	3.550000e+05	76691.200000	90.000000	1.597500e+09
Cloud Data Engineer	122.000000	2021.000000	1.400000e+05	124647.000000	75.000000	6.300000e+08
Computer Vision Engineer	274.833333	2021.166667	8.350000e+04	44419.333333	58.333333	3.757500e+08
Computer Vision Software Engineer	235.666667	2021.333333	1.003333e+05	105248.666667	100.000000	4.515000e+08
Data Analyst	362.010309	2021.680412	9.660496e+04	92893.061856	75.257732	4.347223e+08
Data Analytics Engineer	216.750000	2021.250000	6.175000e+04	64799.250000	75.000000	2.778750e+08
Data Analytics Lead	523.000000	2022.000000	4.050000e+05	405000.000000	100.000000	1.822500e+09
Data Analytics Manager	366.285714	2021.571429	1.271343e+05	127134.285714	85.714286	5.721043e+08
Data Architect	390.636364	2021.727273	1.778739e+05	177873.909091	100.000000	8.004326e+08
Data Engineer	343.537879	2021.590909	1.792106e+05	112725.000000	75.000000	8.064475e+08
Data Engineering Manager	107.200000	2020.600000	1.197998e+05	123227.200000	70.000000	5.390991e+08
Data Science Consultant	138.000000	2020.714286	1.227143e+05	69420.714286	71.428571	5.522143e+08
Data Science Engineer	229.666667	2021.333333	8.450000e+04	75803.333333	83.333333	3.802500e+08

job_title	Unnamed: 0	work_year	salary	salary_in_usd	remote_ratio	salario en pesos
Data Science Manager	274.000000	2021.333333	1.062599e+06	158328.500000	83.333333	4.781694e+09
Data Scientist	314.832168	2021.391608	5.083472e+05	108187.832168	63.986014	2.287562e+09
Data Specialist	165.000000	2021.000000	1.650000e+05	165000.000000	100.000000	7.425000e+08
Director of Data Engineering	171.500000	2021.000000	1.412500e+05	156738.000000	100.000000	6.356250e+08
Director of Data Science	185.857143	2021.000000	1.932857e+05	195074.000000	42.857143	8.697857e+08
ETL Developer	373.500000	2022.000000	5.000000e+04	54957.000000	0.000000	2.250000e+08
Finance Data Analyst	183.000000	2021.000000	4.500000e+04	61896.000000	50.000000	2.025000e+08
Financial Data Analyst	279.000000	2021.500000	2.750000e+05	275000.000000	75.000000	1.237500e+09
Head of Data	302.200000	2021.400000	1.564000e+05	160162.600000	90.000000	7.038000e+08
Head of Data Science	270.250000	2021.500000	1.467188e+05	146718.750000	50.000000	6.602344e+08
Head of Machine Learning	384.000000	2022.000000	6.000000e+06	79039.000000	50.000000	2.700000e+10
Lead Data Analyst	64.333333	2020.666667	5.690000e+05	92203.000000	100.000000	2.560500e+09
Lead Data Engineer	145.500000	2020.833333	1.403333e+05	139724.500000	66.666667	6.315000e+08
Lead Data Scientist	53.000000	2020.333333	1.101667e+06	115190.000000	50.000000	4.957500e+09
Lead Machine Learning Engineer	457.000000	2022.000000	8.000000e+04	87932.000000	0.000000	3.600000e+08
ML Engineer	179.333333	2021.000000	2.676667e+06	117504.000000	83.333333	1.204500e+10
Machine Learning Developer	358.000000	2021.666667	1.000000e+05	85860.666667	83.333333	4.500000e+08
Machine Learning Engineer	288.585366	2021.317073	2.727179e+05	104880.146341	67.073171	1.227230e+09
Machine Learning Infrastructure Engineer	234.333333	2021.000000	9.733333e+04	101145.000000	50.000000	4.380000e+08
Machine Learning Manager	29.000000	2020.000000	1.570000e+05	117104.000000	50.000000	7.065000e+08

job_title	Unnamed: 0	work_year	salary	salary_in_usd	remote_ratio	salario en pesos
Machine Learning Scientist	248.000000	2021.250000	1.584125e+05	158412.500000	68.750000	7.128562e+08
Marketing Data Analyst	90.000000	2021.000000	7.500000e+04	88654.000000	100.000000	3.375000e+08
NLP Engineer	455.000000	2022.000000	2.400000e+05	37236.000000	50.000000	1.080000e+09
Principal Data Analyst	370.000000	2021.500000	1.225000e+05	122500.000000	100.000000	5.512500e+08
Principal Data Engineer	196.000000	2021.000000	3.283333e+05	328333.333333	100.000000	1.477500e+09
Principal Data Scientist	205.285714	2021.000000	2.067143e+05	215242.428571	85.714286	9.302143e+08
Product Data Analyst	12.000000	2020.000000	2.350000e+05	13036.000000	50.000000	1.057500e+09
Research Scientist	246.562500	2021.125000	1.104937e+05	109019.500000	53.125000	4.972216e+08
Staff Data Scientist	283.000000	2021.000000	1.050000e+05	105000.000000	100.000000	4.725000e+08

In [33]:
```
df.groupby("job_title").mean(numeric_only=True).count() #cuenta
```

Out[33]:
```
Unnamed: 0          50
work_year           50
salary              50
salary_in_usd       50
remote_ratio        50
salario en pesos    50
dtype: int64
```

In [36]:
```
df.groupby("job_title").agg({
    "salary": ["max", "mean"]
})  #agrupar por una columan y determinadas medidas
```

Out[36]:

	salary	
	max	mean
job_title		
3D Computer Vision Researcher	400000	4.000000e+05
AI Scientist	1335000	2.905714e+05
Analytics Engineer	205300	1.750000e+05
Applied Data Scientist	380000	1.724000e+05
Applied Machine Learning Scientist	423000	1.413500e+05
BI Data Analyst	11000000	1.902045e+06
Big Data Architect	125000	1.250000e+05
Big Data Engineer	1672000	4.550000e+05
Business Data Analyst	1400000	3.550000e+05
Cloud Data Engineer	160000	1.400000e+05
Computer Vision Engineer	180000	8.350000e+04
Computer Vision Software Engineer	150000	1.003333e+05
Data Analyst	450000	9.660496e+04
Data Analytics Engineer	110000	6.175000e+04
Data Analytics Lead	405000	4.050000e+05
Data Analytics Manager	150260	1.271343e+05
Data Architect	266400	1.778739e+05
Data Engineer	4450000	1.792106e+05
Data Engineering Manager	174000	1.197998e+05
Data Science Consultant	423000	1.227143e+05

	salary	
	max	mean
job_title		
Data Science Engineer	159500	8.450000e+04
Data Science Manager	7000000	1.062599e+06
Data Scientist	30400000	5.083472e+05
Data Specialist	165000	1.650000e+05
Director of Data Engineering	200000	1.412500e+05
Director of Data Science	325000	1.932857e+05
ETL Developer	50000	5.000000e+04
Finance Data Analyst	45000	4.500000e+04
Financial Data Analyst	450000	2.750000e+05
Head of Data	235000	1.564000e+05
Head of Data Science	224000	1.467188e+05
Head of Machine Learning	6000000	6.000000e+06
Lead Data Analyst	1450000	5.690000e+05
Lead Data Engineer	276000	1.403333e+05
Lead Data Scientist	3000000	1.101667e+06
Lead Machine Learning Engineer	80000	8.000000e+04
ML Engineer	8500000	2.676667e+06
Machine Learning Developer	100000	1.000000e+05
Machine Learning Engineer	4900000	2.727179e+05
Machine Learning Infrastructure Engineer	195000	9.733333e+04

	salary	
	max	mean
job_title		
Machine Learning Manager	157000	1.570000e+05
Machine Learning Scientist	260000	1.584125e+05
Marketing Data Analyst	75000	7.500000e+04
NLP Engineer	240000	2.400000e+05
Principal Data Analyst	170000	1.225000e+05
Principal Data Engineer	600000	3.283333e+05
Principal Data Scientist	416000	2.067143e+05
Product Data Analyst	450000	2.350000e+05
Research Scientist	450000	1.104937e+05
Staff Data Scientist	105000	1.050000e+05

contar elementos de una columnas

In [37]:
```
df.shape #tamaño de data
```

Out[37]: (607, 13)

elementos unicos de cada columna

In [38]:
```
df.nunique()
```

Out[38]:
```
Unnamed: 0           607
work_year              3
experience_level       4
employment_type        4
job_title             50
salary               272
salary_currency       17
salary_in_usd        369
employee_residence    57
remote_ratio           3
company_location      50
company_size           3
salario en pesos     272
dtype: int64
```

hacer limpieza de datos

In [39]:
```
df.count() #contar datos
```

Out[39]:
```
Unnamed: 0           607
work_year            607
experience_level     607
employment_type      607
job_title            607
salary               607
salary_currency      607
salary_in_usd        607
employee_residence   607
remote_ratio         607
company_location     607
company_size         607
salario en pesos     607
dtype: int64
```

In [40]:
```
df.isnull().sum() #que datos son nulos
```

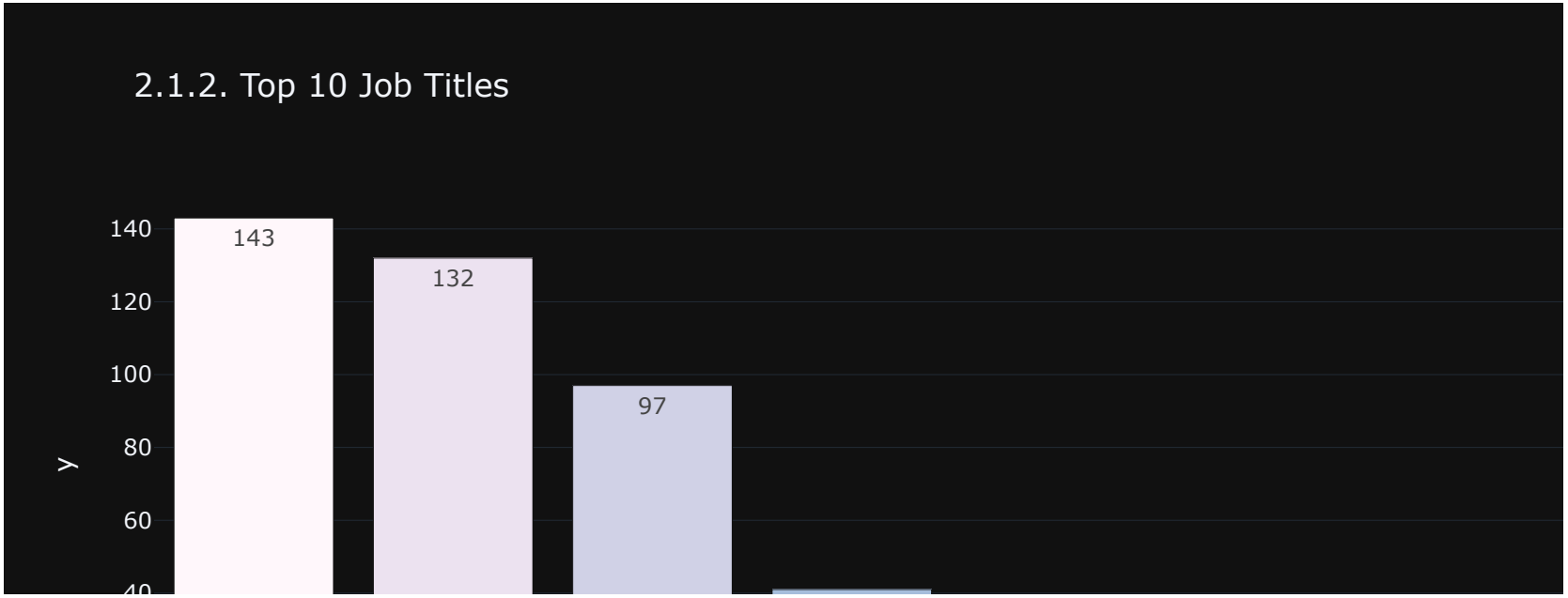
Out[40]:
```
Unnamed: 0           0
work_year            0
experience_level     0
employment_type      0
job_title            0
salary               0
salary_currency      0
salary_in_usd        0
employee_residence   0
remote_ratio         0
company_location     0
company_size         0
salario en pesos     0
dtype: int64
```

Visualizacion de la data a partir de gráficos

In [41]:
```
top10_job_title = df['job_title'].value_counts()[:10] #las primeras 10 empleos mas comunes
```

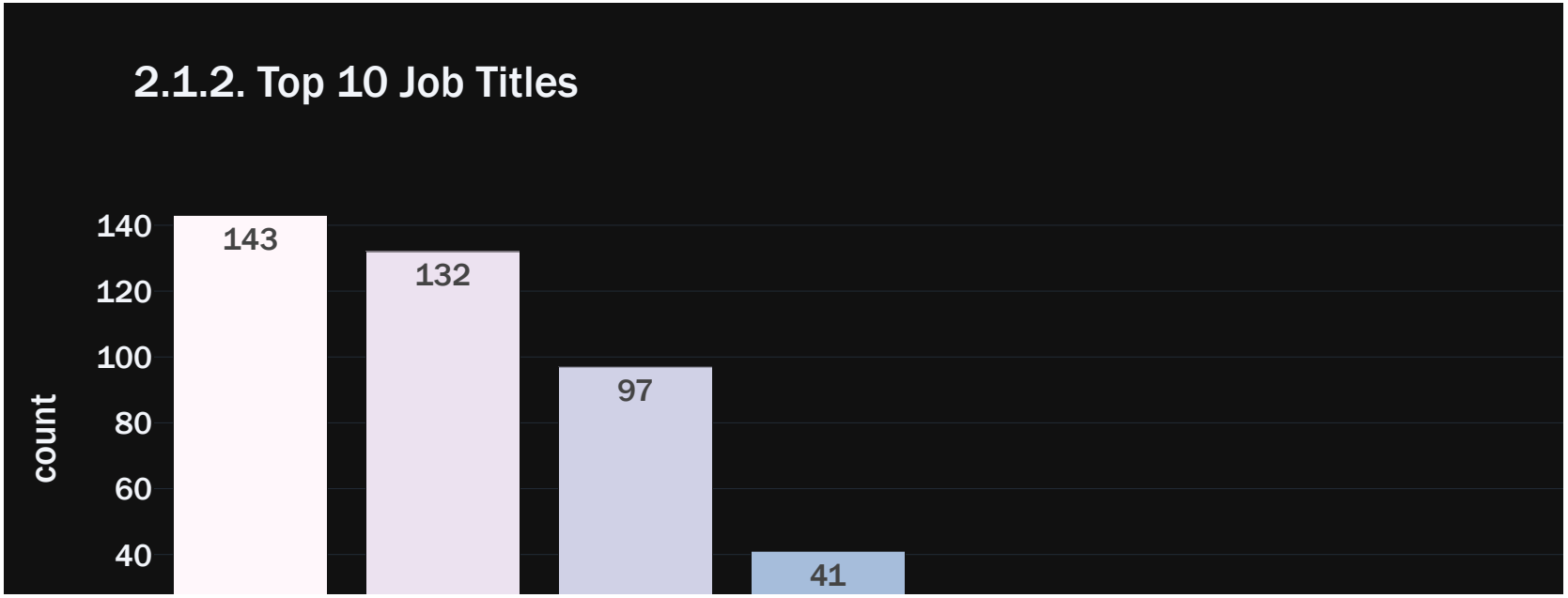
dibujar un diagrama de barras * px.bar(...): Crea un gráfico de barras. * x=top10_job_title.index: Usa los títulos de trabajo (índices de la serie) como el eje X. * y=top10_job_title.values: Usa la cantidad de veces que aparecen los títulos como eje Y. * color=top10_job_title.index: Asigna diferentes colores a cada categoría (título de trabajo). * color_discrete_sequence=px.colors.sequential.PuBuGn: Usa una paleta de colores predefinida (PuBuGn). * text=top10_job_title.values: Muestra los valores sobre las barras. * title='2.1.2. Top 10 Job Titles': Agrega un título al gráfico. * template='plotly_dark': Usa un tema oscuro para el diseño.

In [42]:
```
fig = px.bar(y=top10_job_title.values,
             x=top10_job_title.index,
             color = top10_job_title.index,
             color_discrete_sequence=px.colors.sequential.PuBuGn,
             text=top10_job_title.values,
             title= '2.1.2. Top 10 Job Titles',
             template= 'plotly_dark')
fig.show()
```



El método update_layout() se usa para modificar el diseño del gráfico. Aquí está lo que hace cada argumento: * xaxis_title="Job Titles" : Cambia el título del eje X a "Job Titles" (Títulos de Trabajo). y Este eje representa las categorías (diferentes títulos de trabajo). *yaxis_title="count" : Cambia el título del eje Y a "count" (Cantidad). Este eje muestra la frecuencia de cada título de trabajo en los datos. * font=dict(size=17, family="Franklin Gothic") Ajusta el tamaño y la fuente del texto en el gráfico. size=17: Aumenta el tamaño del texto a 17 puntos. family="Franklin Gothic": Usa la fuente "Franklin Gothic" para los textos.

In [43]:
```
fig.update_layout(
    xaxis_title="Job Titles",
    yaxis_title="count",
    font = dict(size=17,family="Franklin Gothic"))
fig.show()
```



vamos a construir un digrama de lineas por cada variable cuantitativa, sirve para ver el comportramiento de una variable en el tiempo

In [44]:
```
df_cuant= df.select_dtypes(include=['int64', 'float64'])
df_cuant
```

Out[44]:

	Unnamed: 0	work_year	salary	salary_in_usd	remote_ratio	salario en pesos
0	0	2020	70000	79833	0	315000000
1	1	2020	260000	260000	0	1170000000
2	2	2020	85000	109024	50	382500000
3	3	2020	20000	20000	0	90000000
4	4	2020	150000	150000	50	675000000
...	...	...	...	...	...	...
602	602	2022	154000	154000	100	693000000
603	603	2022	126000	126000	100	567000000
604	604	2022	129000	129000	0	580500000
605	605	2022	150000	150000	100	675000000
606	606	2022	200000	200000	100	900000000

607 rows × 6 columns

In [45]:
```
df_cuant= df_cuant.iloc[:, 1:]
```

In [46]:
```
df_cuant
```

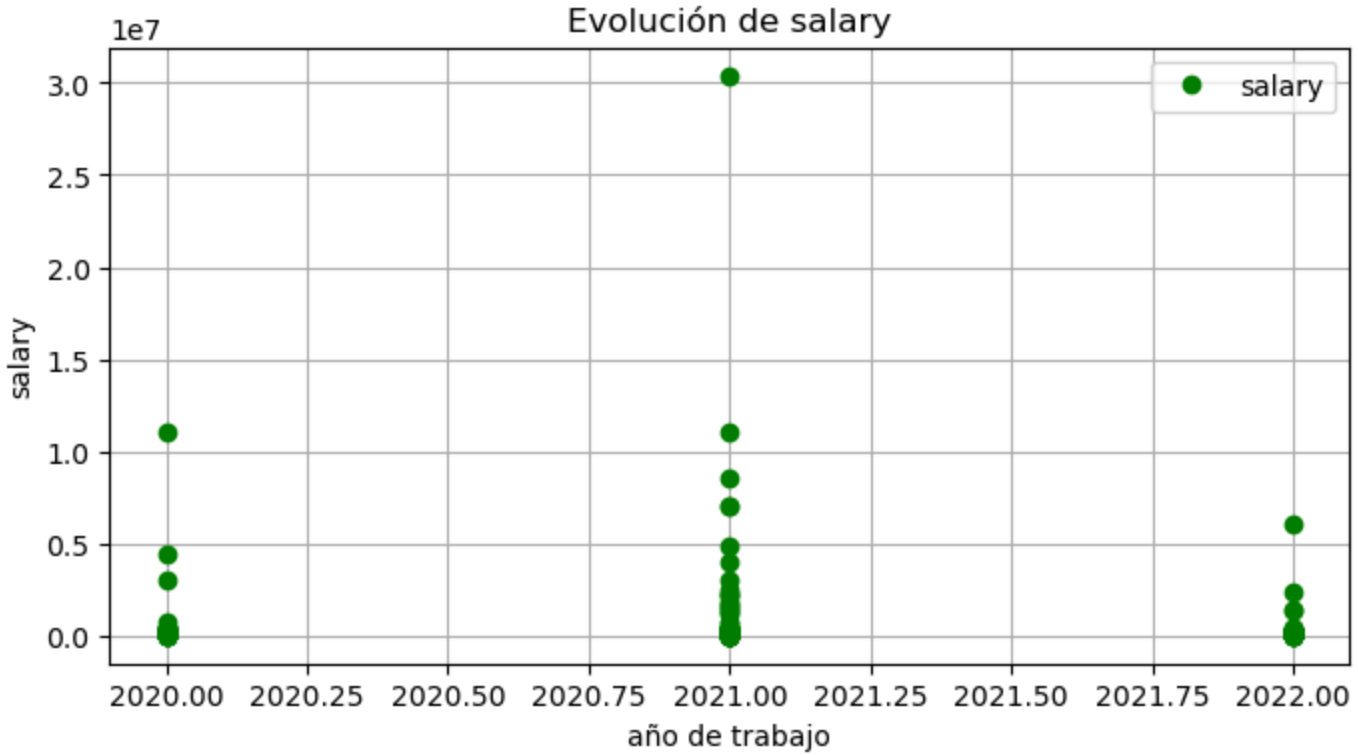
Out[46]:

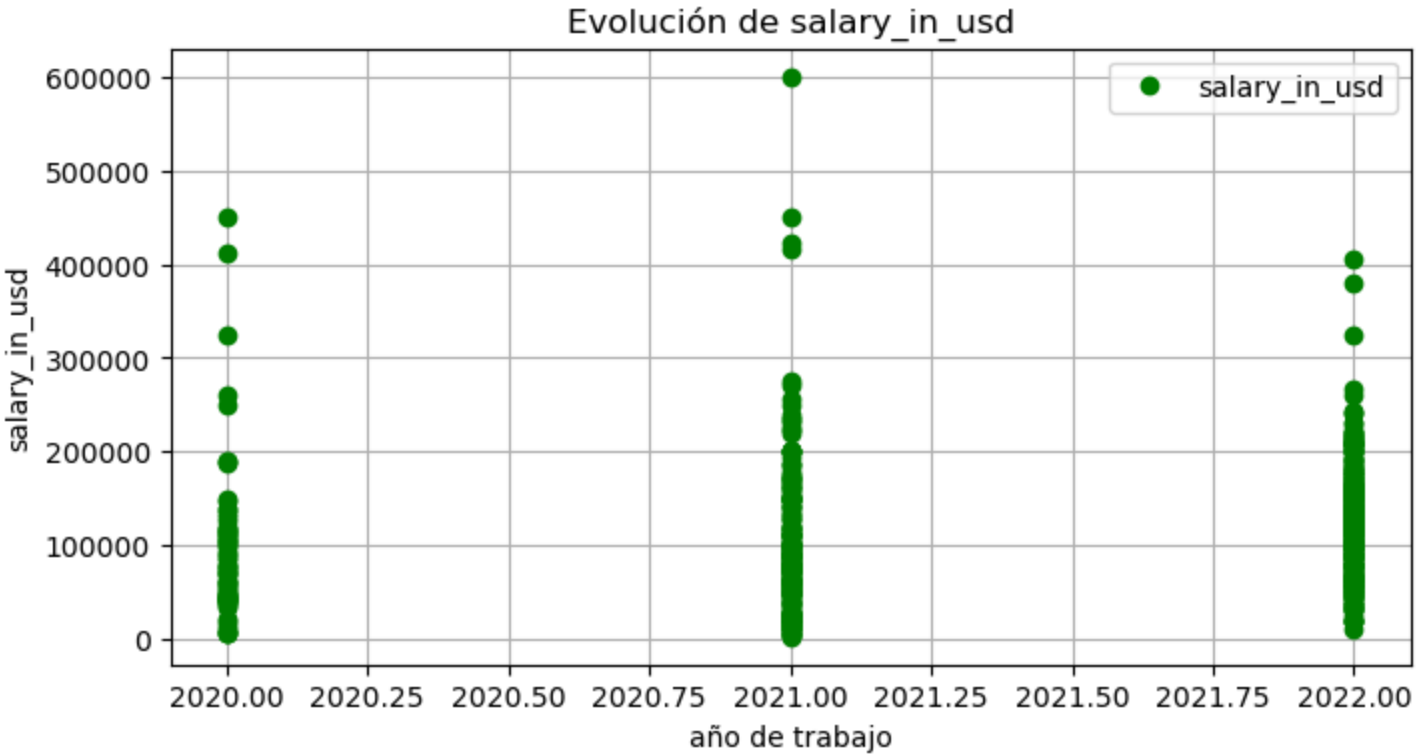
	work_year	salary	salary_in_usd	remote_ratio	salario en pesos
0	2020	70000	79833	0	315000000
1	2020	260000	260000	0	1170000000
2	2020	85000	109024	50	382500000
3	2020	20000	20000	0	90000000
4	2020	150000	150000	50	675000000
...	...	...	...	...	...
602	2022	154000	154000	100	693000000
603	2022	126000	126000	100	567000000
604	2022	129000	129000	0	580500000
605	2022	150000	150000	100	675000000
606	2022	200000	200000	100	900000000

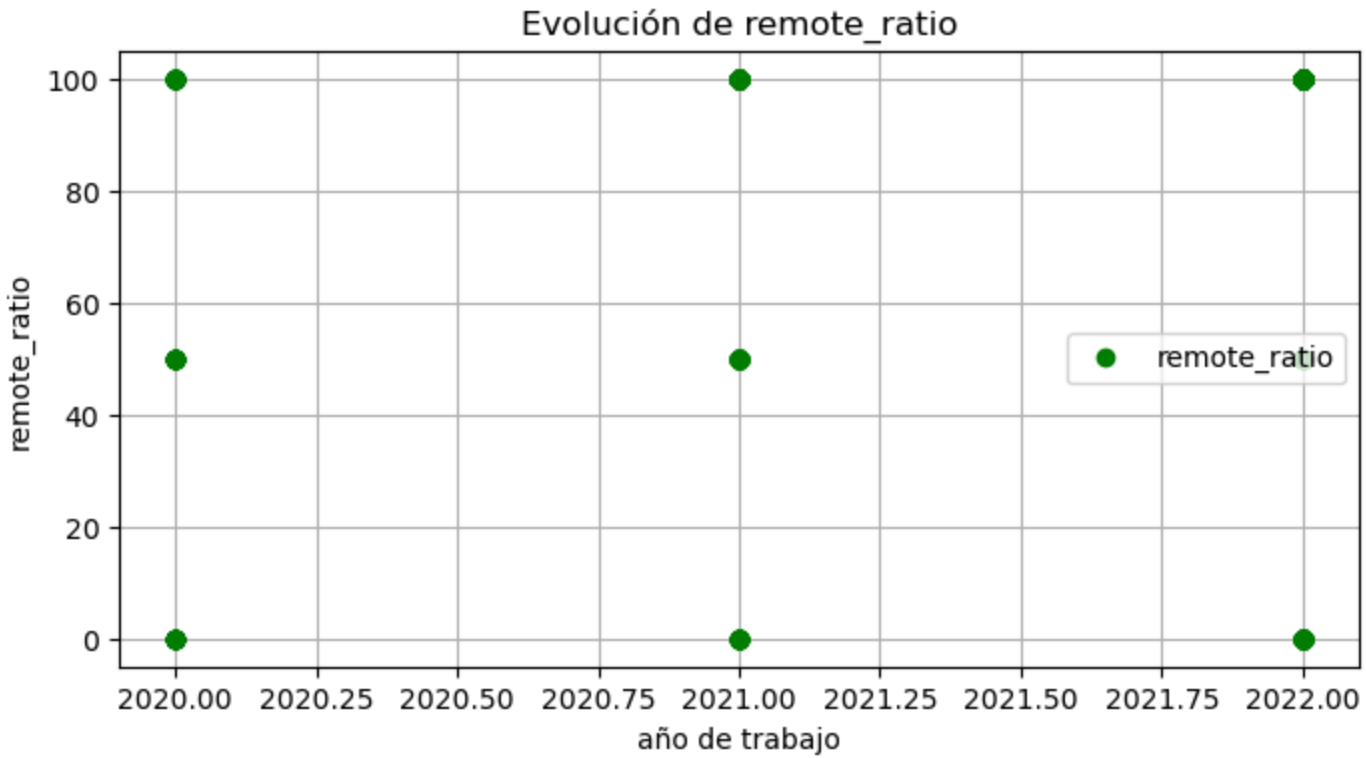
607 rows × 5 columns

Gráficar uno por uno

In [47]:
```
for i in range(1, df_cuant.shape[1]):
    plt.figure(figsize=(8, 4))  # Crear una nueva figura para cada gráfico

    plt.plot(df_cuant.work_year, df_cuant.iloc[:, i], marker="o", linestyle="",color="green", label=df_cuant.columns

    # Personalización del gráfico
    plt.xlabel("año de trabajo")
    plt.ylabel(df_cuant.columns[i])
    plt.title(f"Evolución de {df_cuant.columns[i]}")
    plt.legend()
    plt.grid(True)

    plt.show()  # Mostrar cada gráf
```


Evolución de salary
1e7
3.0
2.5
2.0
1.5
1.0
0.5
0.0
salary
2020.00
2020.25
2020.50
2020.75
2021.00
2021.25
2021.50
2021.75
2022.00
año de trabajo
salary


Evolución de salary_in_usd
salary_in_usd
600000
500000
400000
300000
200000
100000
0
salary_in_usd
2020.00
2020.25
2020.50
2020.75
2021.00
2021.25
2021.50
2021.75
2022.00
año de trabajo


Evolución de remote_ratio
remote_ratio
100
80
60
40
20
0
2020.00
2020.25
2020.50
2020.75
2021.00
2021.25
2021.50
2021.75
2022.00
año de trabajo
remote_ratio

distribución normal

In [48]:
```
df_cuant= df_cuant.iloc[:,1:]
df_cuant
```

Out[48]:

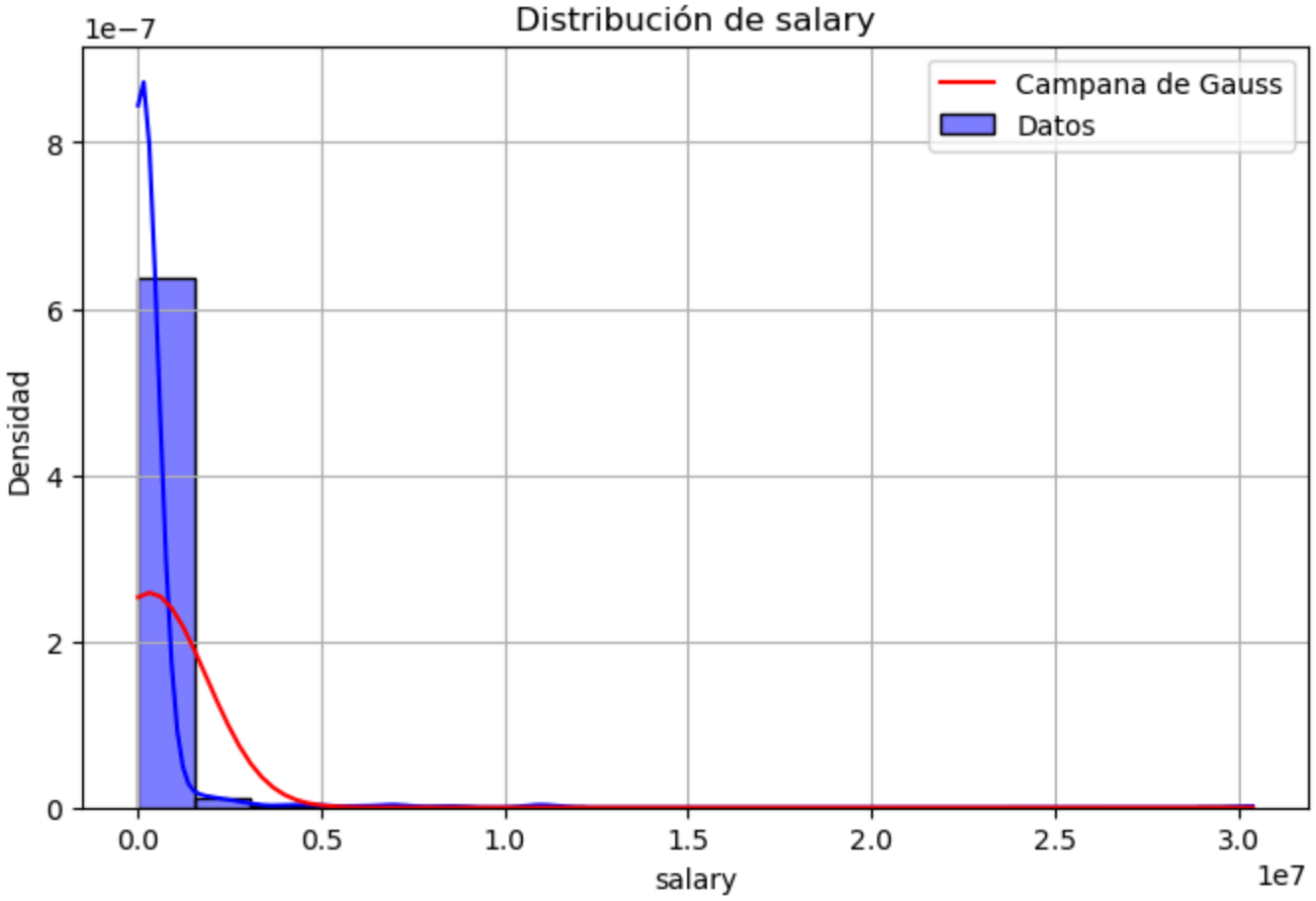
	salary	salary_in_usd	remote_ratio	salario en pesos
0	70000	79833	0	315000000
1	260000	260000	0	1170000000
2	85000	109024	50	382500000
3	20000	20000	0	90000000
4	150000	150000	50	675000000
...	...	...	...	...
602	154000	154000	100	693000000
603	126000	126000	100	567000000
604	129000	129000	0	580500000
605	150000	150000	100	675000000
606	200000	200000	100	900000000

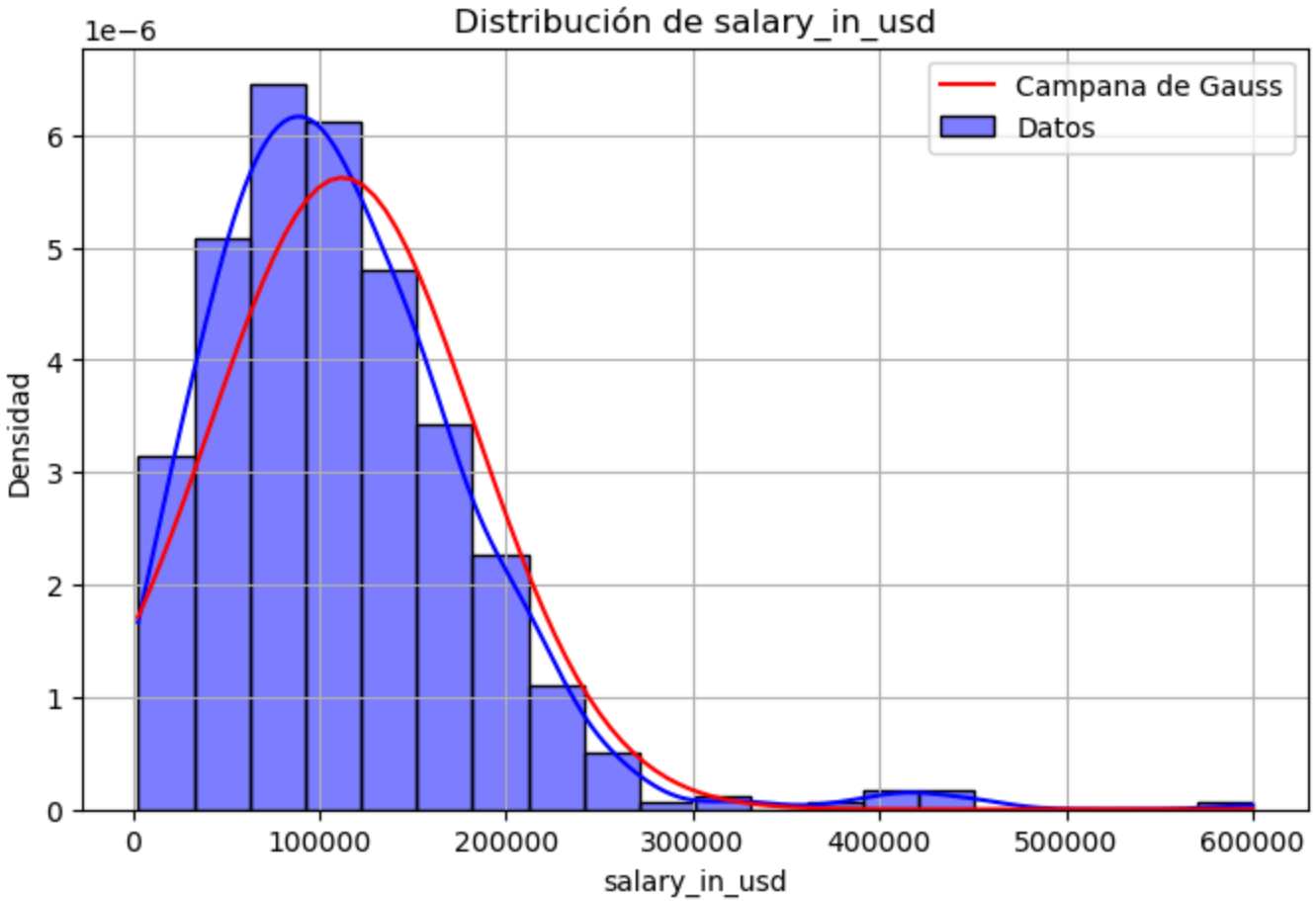
607 rows × 4 columns

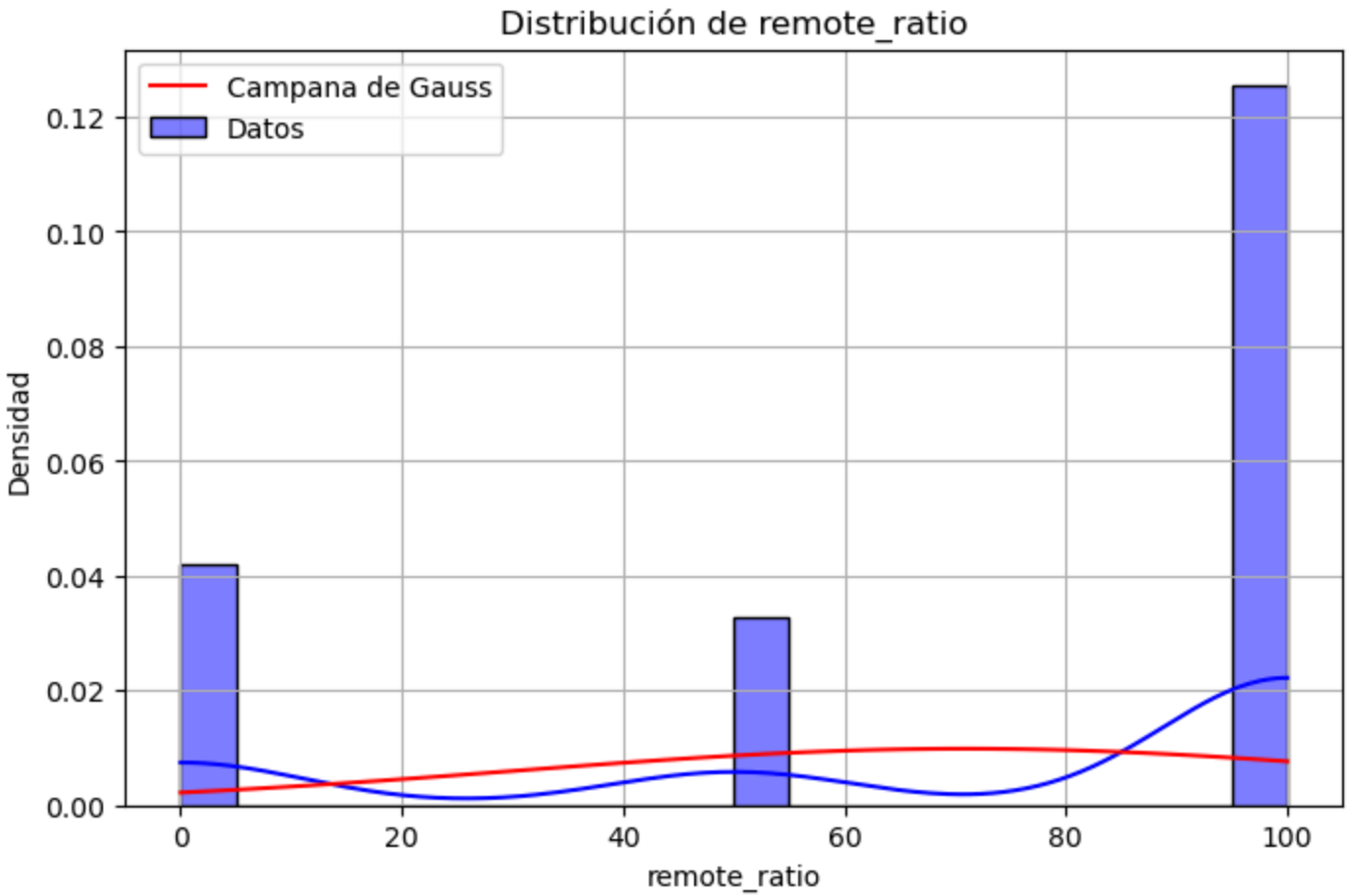
distribución normal de los datos

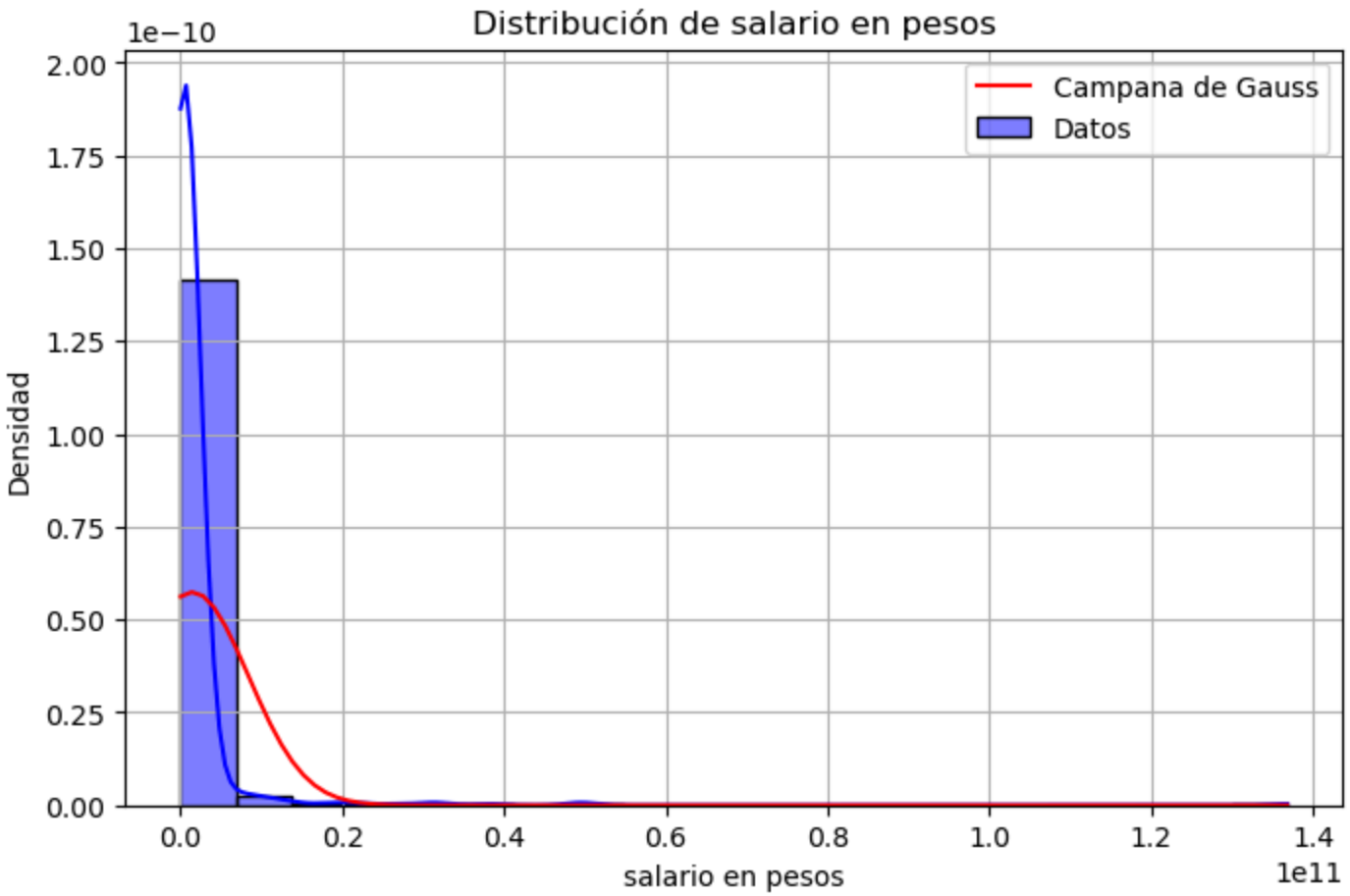
In [49]:
```
# Graficar cada variable numérica con su campana de Gauss
for columna in df_cuant.columns:
    plt.figure(figsize=(8, 5))  # Nueva figura para cada variable

    # Histograma con densidad
    sns.histplot(df_cuant[columna], kde=True, bins=20, stat="density", color="blue", label="Datos")

    # Ajuste de la curva normal teórica
    media = df_cuant[columna].mean()
    desviacion = df_cuant[columna].std()
    x = np.linspace(df_cuant[columna].min(), df_cuant[columna].max(), 100) #linea de ahuste de la cuerva
    y = norm.pdf(x, media, desviacion)
    plt.plot(x, y, color="red", label="Campana de Gauss")

    # Personalización del gráfico
    plt.title(f"Distribución de {columna}")
```

```
    plt.xlabel(columna)
    plt.ylabel("Densidad")
    plt.legend()
    plt.grid(True)

    plt.show()  # Muestra cada gráfico individualmente
```




Distribución de salary_in_usd
1e−6
Campana de Gauss
Datos
Densidad
salary_in_usd
0
1
2
3
4
5
6
0
100000
200000
300000
400000
500000
600000


Distribución de remote_ratio
Campana de Gauss
Datos
Densidad
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0
20
40
60
80
100
remote_ratio



la correlacción entre los datos, sirve para revisar la relacion de los datos

In [50]:
```
correlacion = df_cuant.corr()
```

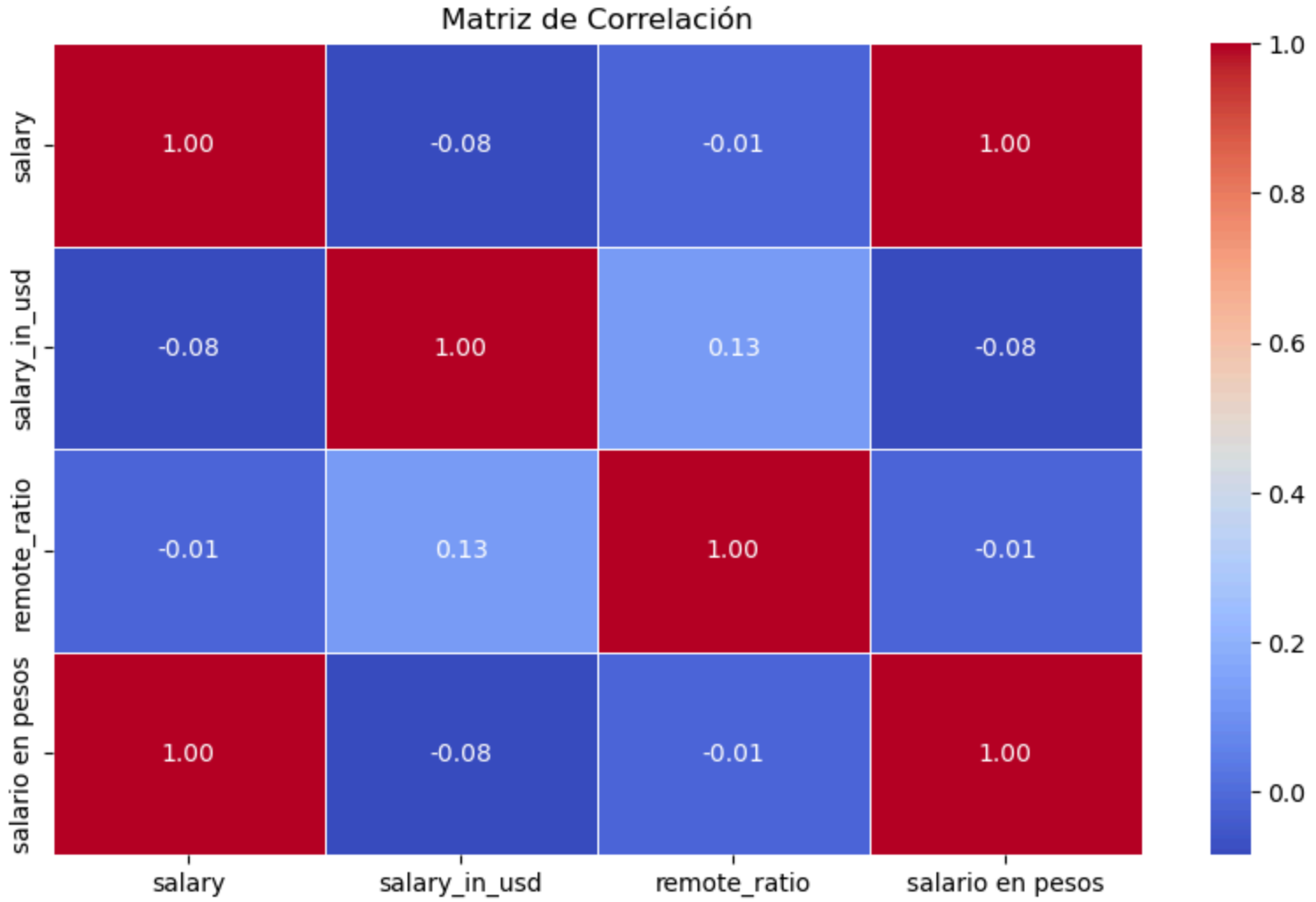
In [51]:
```
correlacion
```

Out[51]:

	salary	salary_in_usd	remote_ratio	salario en pesos
salary	1.000000	-0.083906	-0.014608	1.000000
salary_in_usd	-0.083906	1.000000	0.132122	-0.083906
remote_ratio	-0.014608	0.132122	1.000000	-0.014608
salario en pesos	1.000000	-0.083906	-0.014608	1.000000

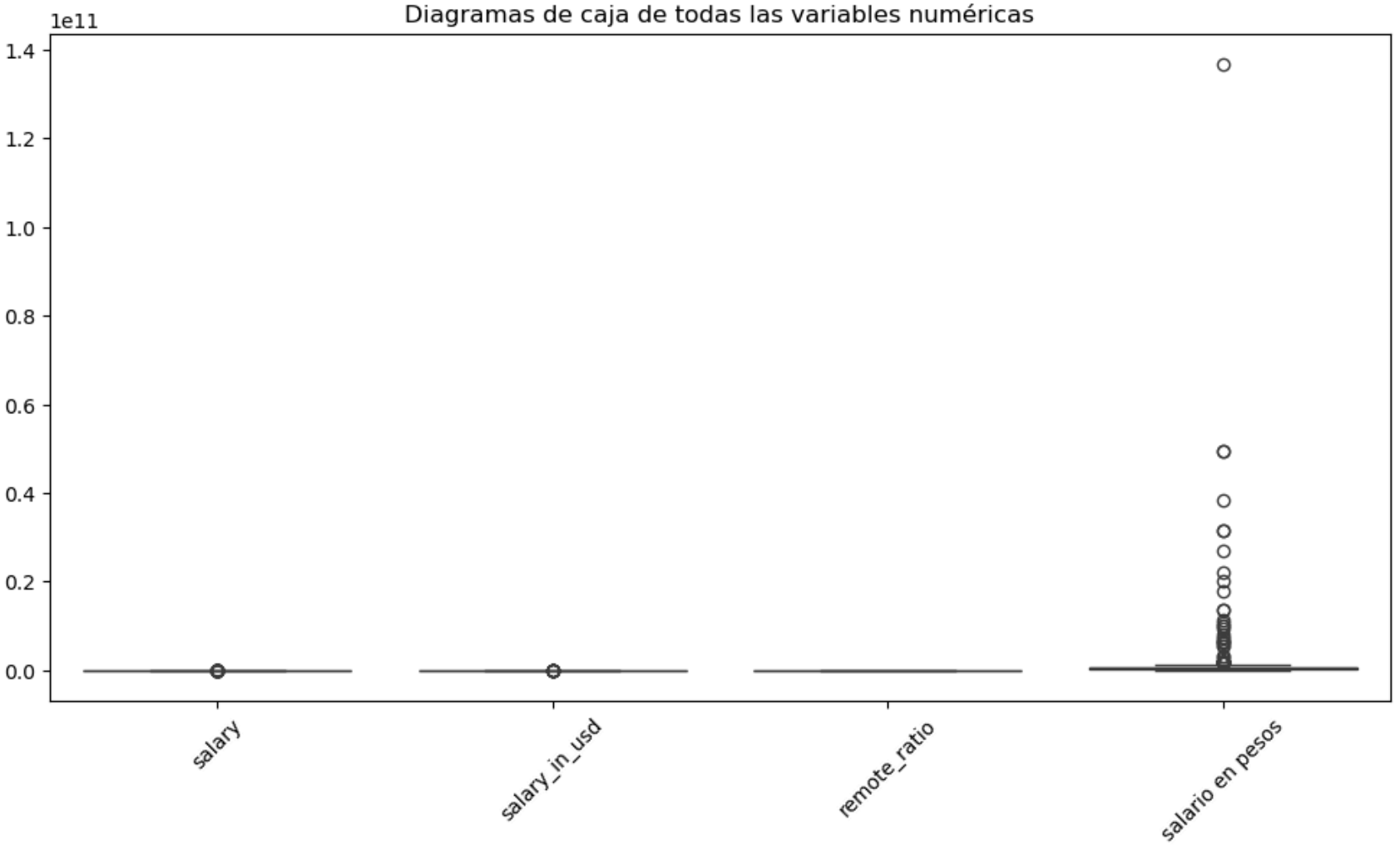
In [52]:

```
correlacion = df_cuant.corr()
#  🔹 Crear el mapa de calor
plt.figure(figsize=(10, 6))  # Ajustar tamaño de la figura
sns.heatmap(correlacion, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)

#  🔹 Título del gráfico
plt.title("Matriz de Correlación")

#  🔹 Mostrar el gráfico
plt.show()
```



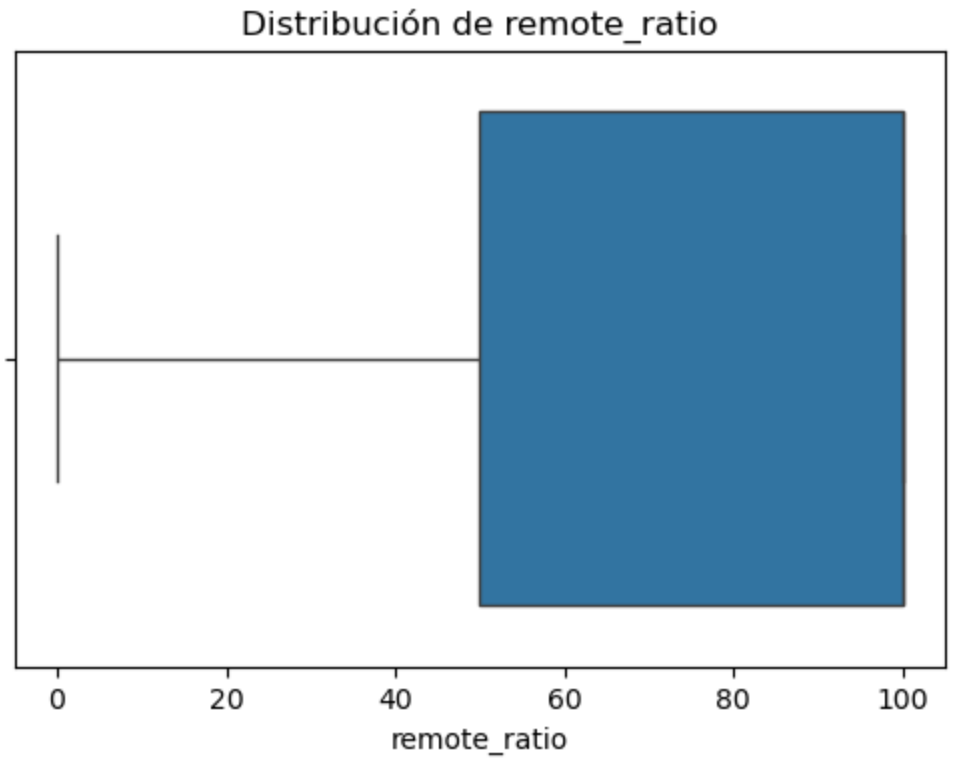
In [53]:
```
import seaborn as sns
import matplotlib.pyplot as plt

#  🔹 Seleccionar solo las columnas numéricas del DataFrame


#  🔹 Crear un boxplot para todas las columnas numéricas
plt.figure(figsize=(12,6))  # Tamaño del gráfico
sns.boxplot(df_cuant)
```

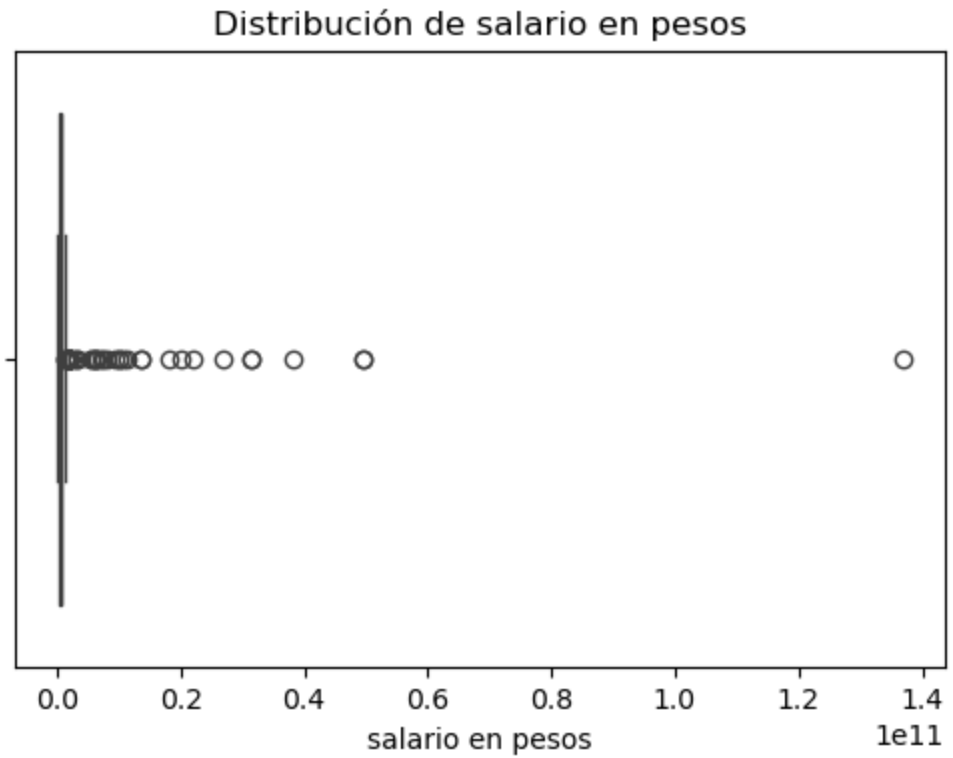
```
#  🔹 Mejorar visualización
plt.xticks(rotation=45)  # Rotar nombres de variables
plt.title("Diagramas de caja de todas las variables numéricas")

#  🔹 Mostrar gráfico
plt.show()
```



In [54]:
```
#  🔹 Recorrer cada columna numérica y hacer un boxplot individual
for i in range(1, df_cuant.shape[1]):
    plt.figure(figsize=(6,4))  # Tamaño de cada gráfico
```

```
    sns.boxplot(x=df_cuant.iloc[:, i])
    plt.title(f"Distribución de {df_cuant.columns[i]}")  # Título con el nombre de la variable
    plt.show()
```


Distribución de remote_ratio
0
20
40
60
80
100
remote_ratio



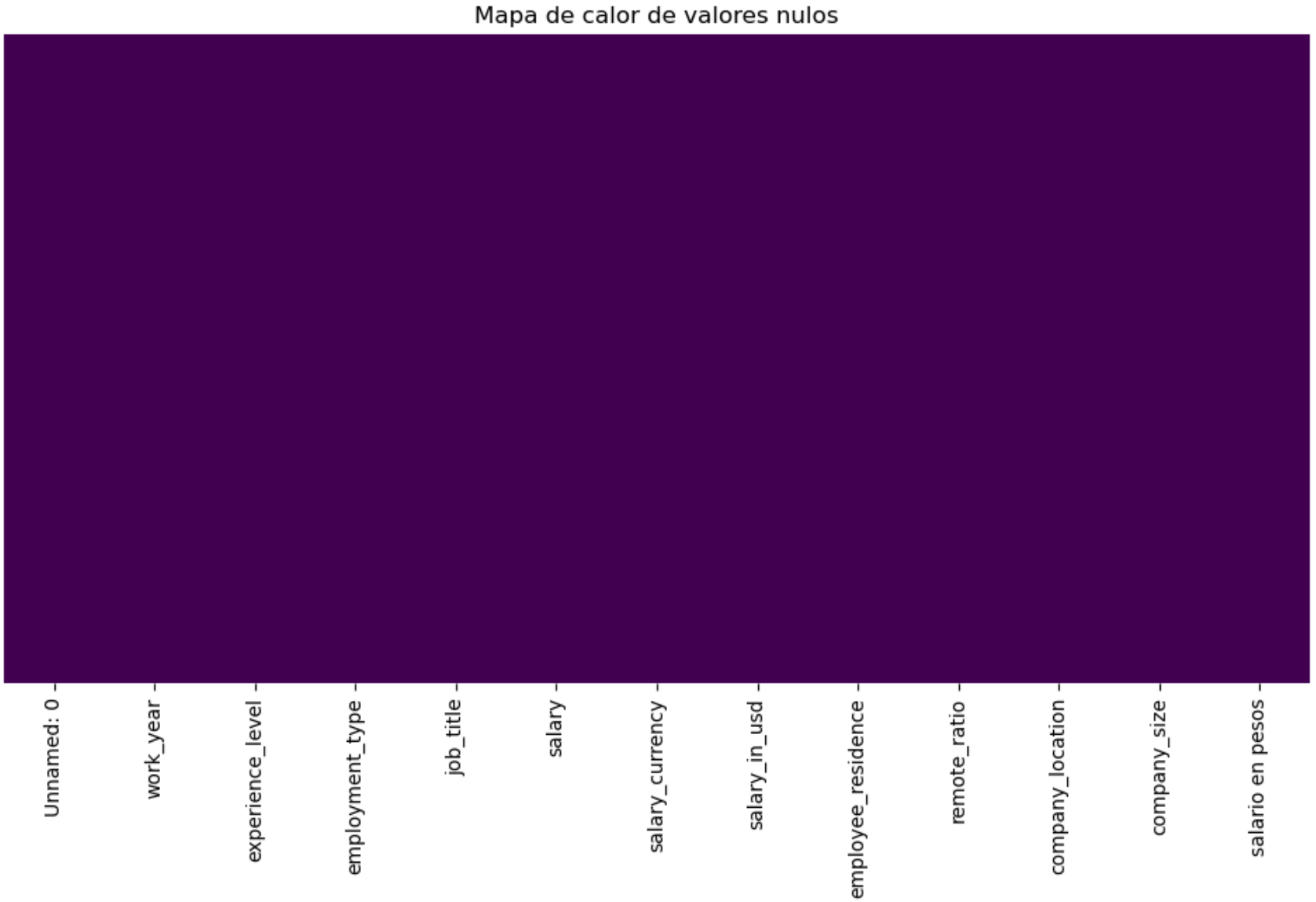
In [55]:
```
df_cuant.describe()
```

Out[55]:

	salary	salary_in_usd	remote_ratio	salario en pesos
count	6.070000e+02	607.000000	607.00000	6.070000e+02
mean	3.240001e+05	112297.869852	70.92257	1.458000e+09
std	1.544357e+06	70957.259411	40.70913	6.949609e+09
min	4.000000e+03	2859.000000	0.00000	1.800000e+07
25%	7.000000e+04	62726.000000	50.00000	3.150000e+08
50%	1.150000e+05	101570.000000	100.00000	5.175000e+08
75%	1.650000e+05	150000.000000	100.00000	7.425000e+08
max	3.040000e+07	600000.000000	100.00000	1.368000e+11

valores nulos en la data

In [56]:
```
plt.figure(figsize=(12,6))
sns.heatmap(df.isnull(), cmap="viridis", cbar=False, yticklabels=False)
plt.title("Mapa de calor de valores nulos")
plt.show()
```

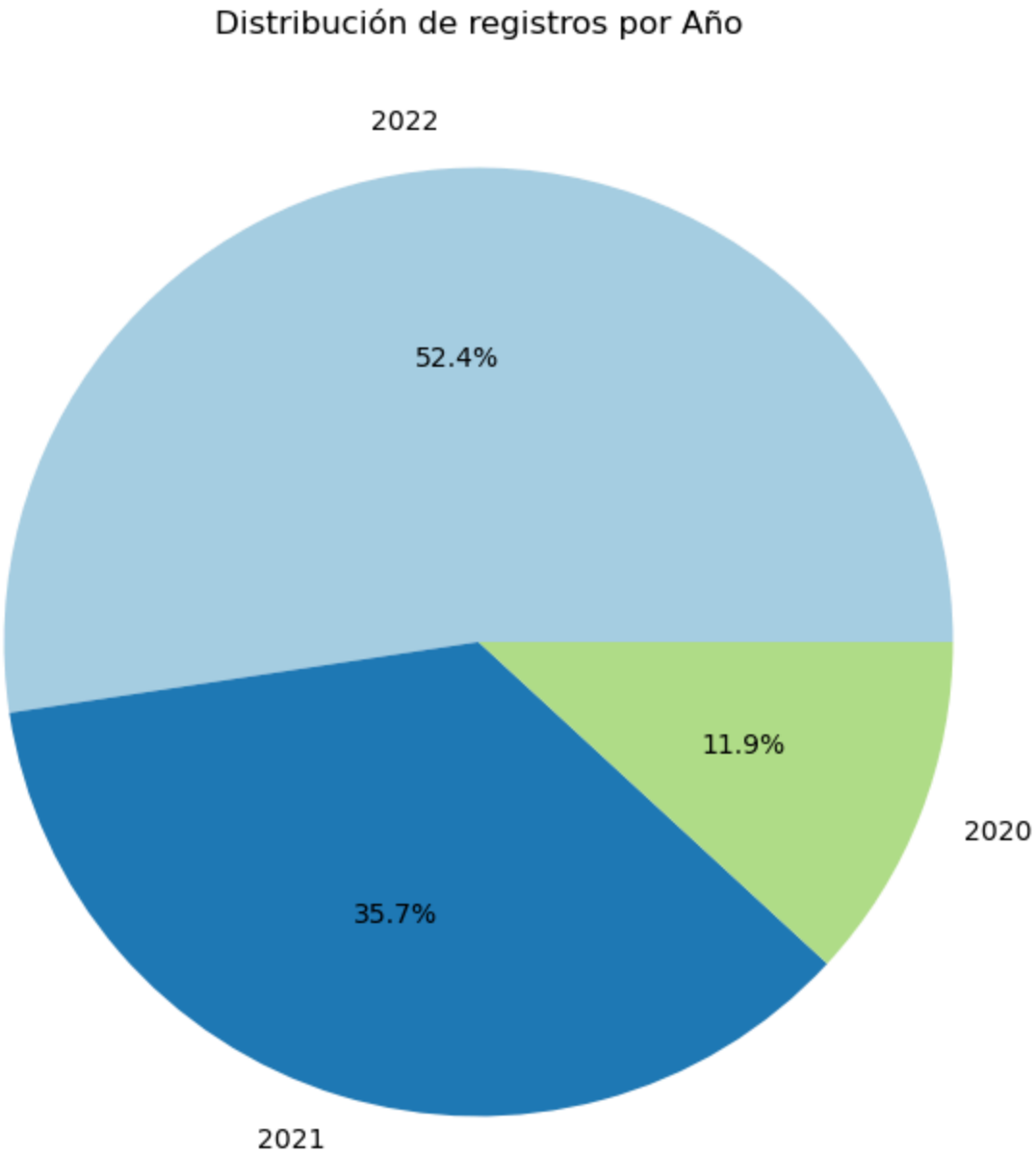


los espacios en blanco son nulos

In [57]:
```
plt.figure(figsize=(12,6))
sns.heatmap(df_cuant.isnull(), cmap="viridis", cbar=False, yticklabels=False)
plt.title("Mapa de calor de valores nulos")
plt.show()
```

In [58]:
```
# Contar cuántos registros hay por año
conteo_años = df["work_year"].value_counts()
print(conteo_años)

# Crear el gráfico de torta
plt.figure(figsize=(8,8))
plt.pie(conteo_años, labels=conteo_años.index, autopct="%1.1f%%", colors=plt.cm.Paired.colors)

# Título y mostrar gráfico
plt.title("Distribución de registros por Año")
plt.show()
```

```
work_year
2022    318
2021    217
2020     72
Name: count, dtype: int64
```


Distribución de registros por Año
2022
52.4%
11.9%
2020
35.7%
2021

Últimas Exploraciones de la data para aplicar modelos

Ya vimos que si aplicamos una regresión lineal no será el mejor de los resultados, porque analizamos con solo variables cuantitativas, ahora vamos a medir con variables cualitativas de caracter ordinal (la unica que se puede). Entonces tenemos que convertir datos categoricos en números.

In [59]:
```
df["experiencia_num"]= df["experience_level"].replace({'EN': 1, 'MI': 2, 'SE': 3, 'EX': 4})
df
```

```
C:\Users\darly\AppData\Local\Temp\ipykernel_21952\4027909897.py:1: FutureWarning:

Downcasting behavior in `replace` is deprecated and will be removed in a future version. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
```

Out[59]:

	Unnamed: 0	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_resid
0	0	2020	MI	FT	Data Scientist	70000	EUR	79833	
1	1	2020	SE	FT	Machine Learning Scientist	260000	USD	260000	
2	2	2020	SE	FT	Big Data Engineer	85000	GBP	109024	
3	3	2020	MI	FT	Product Data Analyst	20000	USD	20000	
4	4	2020	SE	FT	Machine Learning Engineer	150000	USD	150000	
...	...	...	...	...	...	...	...	...	
602	602	2022	SE	FT	Data Engineer	154000	USD	154000	
603	603	2022	SE	FT	Data Engineer	126000	USD	126000	
604	604	2022	SE	FT	Data Analyst	129000	USD	129000	
605	605	2022	SE	FT	Data Analyst	150000	USD	150000	
606	606	2022	MI	FT	AI Scientist	200000	USD	200000	

607 rows × 14 columns

In [60]:
```
df["tamanio_campania"]= df["company_size"].replace({'S': 1, 'M': 2, 'L': 3})
```

```
df
```

```
C:\Users\darly\AppData\Local\Temp\ipykernel_21952\911041496.py:1: FutureWarning:

Downcasting behavior in `replace` is deprecated and will be removed in a future version. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
```

Out[60]:

	Unnamed: 0	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_resid
0	0	2020	MI	FT	Data Scientist	70000	EUR	79833	
1	1	2020	SE	FT	Machine Learning Scientist	260000	USD	260000	
2	2	2020	SE	FT	Big Data Engineer	85000	GBP	109024	
3	3	2020	MI	FT	Product Data Analyst	20000	USD	20000	
4	4	2020	SE	FT	Machine Learning Engineer	150000	USD	150000	
...	...	...	...	...	...	...	...	...	
602	602	2022	SE	FT	Data Engineer	154000	USD	154000	
603	603	2022	SE	FT	Data Engineer	126000	USD	126000	
604	604	2022	SE	FT	Data Analyst	129000	USD	129000	
605	605	2022	SE	FT	Data Analyst	150000	USD	150000	
606	606	2022	MI	FT	AI Scientist	200000	USD	200000	

607 rows × 15 columns

In [61]:
```
df.columns
```

Out[61]:
```
Index(['Unnamed: 0', 'work_year', 'experience_level', 'employment_type',
       'job_title', 'salary', 'salary_currency', 'salary_in_usd',
       'employee_residence', 'remote_ratio', 'company_location',
       'company_size', 'salario en pesos', 'experiencia_num',
       'tamanio_campania'],
      dtype='object')
```

pronto vamos a almacenar la data que se esta limpiando, entonces eliminamos las columnas que no aportan a la data

In [62]:
```
df= df.drop(columns=["Unnamed: 0", "salario en pesos"], inplace=False) #eliminar columna mal nombrada, y creada
df
```

Out[62]:

	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_residence	remo
0	2020	MI	FT	Data Scientist	70000	EUR	79833	DE	
1	2020	SE	FT	Machine Learning Scientist	260000	USD	260000	JP	
2	2020	SE	FT	Big Data Engineer	85000	GBP	109024	GB	
3	2020	MI	FT	Product Data Analyst	20000	USD	20000	HN	
4	2020	SE	FT	Machine Learning Engineer	150000	USD	150000	US	
...	...	...	...	...	...	...	...	...	
602	2022	SE	FT	Data Engineer	154000	USD	154000	US	
603	2022	SE	FT	Data Engineer	126000	USD	126000	US	
604	2022	SE	FT	Data Analyst	129000	USD	129000	US	
605	2022	SE	FT	Data Analyst	150000	USD	150000	US	
606	2022	MI	FT	AI Scientist	200000	USD	200000	IN	

607 rows × 13 columns

Vamos creando la data que vamos a medirsacar solo las columnas numericas

In [64]:
```
df_analisis= df.select_dtypes(include=['int64', 'float64'])
```

```
df_analisis
```

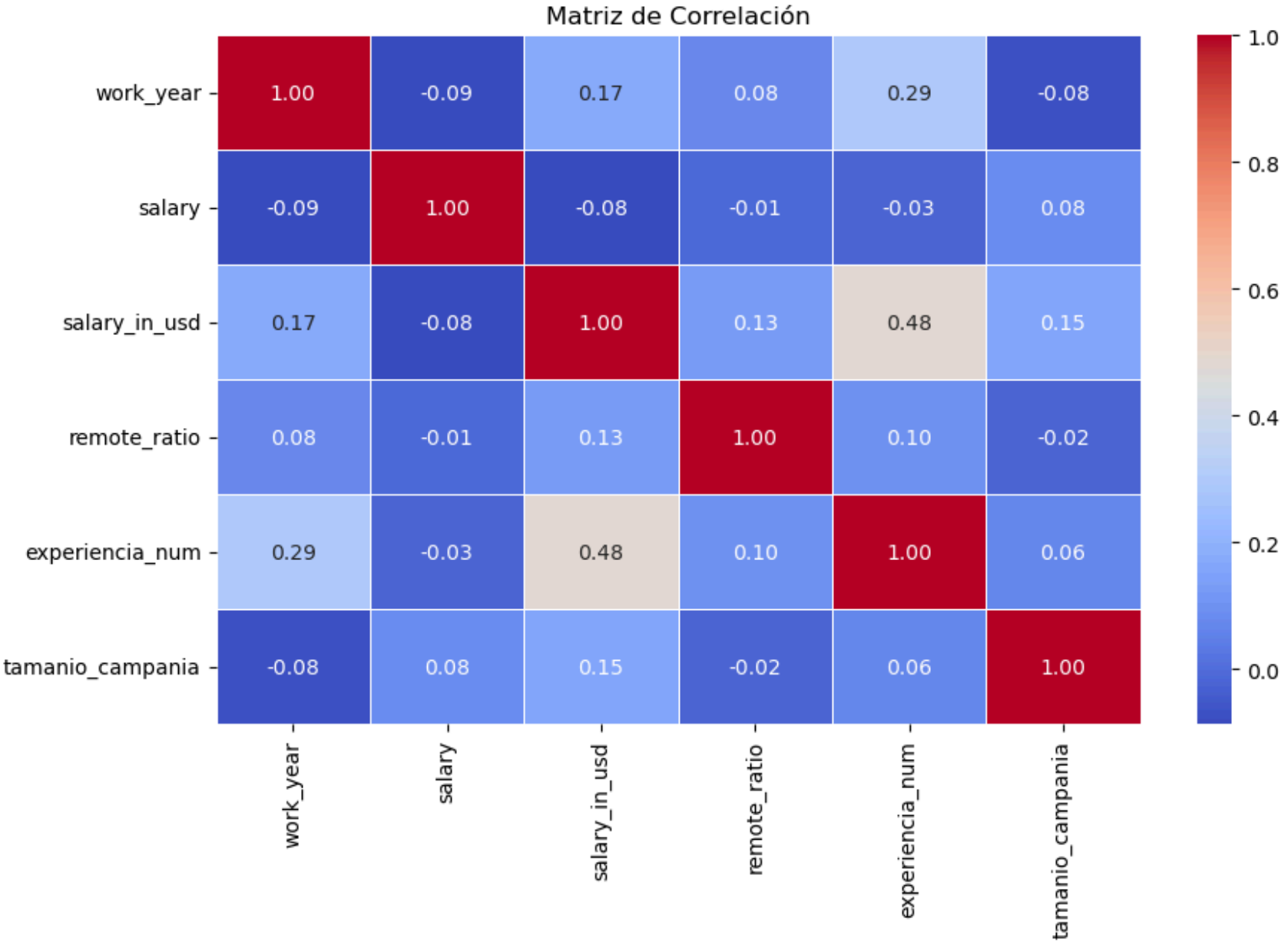
Out[64]:

	work_year	salary	salary_in_usd	remote_ratio	experiencia_num	tamanio_campania
0	2020	70000	79833	0	2	3
1	2020	260000	260000	0	3	1
2	2020	85000	109024	50	3	2
3	2020	20000	20000	0	2	1
4	2020	150000	150000	50	3	3
...	...	...	...	...	...	...
602	2022	154000	154000	100	3	2
603	2022	126000	126000	100	3	2
604	2022	129000	129000	0	3	2
605	2022	150000	150000	100	3	2
606	2022	200000	200000	100	2	3

607 rows × 6 columns

análisis de correlación

In [65]:
```
correlacion = df_analisis.corr()
#  🔹 Crear el mapa de calor
plt.figure(figsize=(10, 6))  # Ajustar tamaño de la figura
sns.heatmap(correlacion, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)

#  🔹 Título del gráfico
plt.title("Matriz de Correlación")

#  🔹 Mostrar el gráfico
plt.show()
```



Ya vimos algo que cambio... vamos a agregar mas elementos a ver si se modifica Revisemos si asociar los empleos pueden jugar un papel importante

In [66]:
```
df['job_title'].nunique() #saca suma de unicos
```

Out[66]: 50

In [67]:
```
job_uni= list(df['job_title'].unique()) #una lista de los valores unicos
job_uni
```

Out[67]:
```
['Data Scientist',
 'Machine Learning Scientist',
 'Big Data Engineer',
 'Product Data Analyst',
 'Machine Learning Engineer',
 'Data Analyst',
 'Lead Data Scientist',
 'Business Data Analyst',
 'Lead Data Engineer',
 'Lead Data Analyst',
 'Data Engineer',
 'Data Science Consultant',
 'BI Data Analyst',
 'Director of Data Science',
 'Research Scientist',
 'Machine Learning Manager',
 'Data Engineering Manager',
 'Machine Learning Infrastructure Engineer',
 'ML Engineer',
 'AI Scientist',
 'Computer Vision Engineer',
 'Principal Data Scientist',
 'Data Science Manager',
 'Head of Data',
 '3D Computer Vision Researcher',
 'Data Analytics Engineer',
 'Applied Data Scientist',
 'Marketing Data Analyst',
 'Cloud Data Engineer',
 'Financial Data Analyst',
 'Computer Vision Software Engineer',
 'Director of Data Engineering',
 'Data Science Engineer',
 'Principal Data Engineer',
 'Machine Learning Developer',
 'Applied Machine Learning Scientist',
 'Data Analytics Manager',
 'Head of Data Science',
 'Data Specialist',
 'Data Architect',
 'Finance Data Analyst',
 'Principal Data Analyst',
```

```
 'Big Data Architect',
 'Staff Data Scientist',
 'Analytics Engineer',
 'ETL Developer',
 'Head of Machine Learning',
 'NLP Engineer',
 'Lead Machine Learning Engineer',
 'Data Analytics Lead']
```

In [68]:
```
#un diccionario con los datos unicos y su respectivo valor
dict_job= {}
for i in range(df['job_title'].nunique()):
    dict_job[job_uni[i]]=i+1

dict_job
```

Out[68]:

```
{'Data Scientist': 1,
 'Machine Learning Scientist': 2,
 'Big Data Engineer': 3,
 'Product Data Analyst': 4,
 'Machine Learning Engineer': 5,
 'Data Analyst': 6,
 'Lead Data Scientist': 7,
 'Business Data Analyst': 8,
 'Lead Data Engineer': 9,
 'Lead Data Analyst': 10,
 'Data Engineer': 11,
 'Data Science Consultant': 12,
 'BI Data Analyst': 13,
 'Director of Data Science': 14,
 'Research Scientist': 15,
 'Machine Learning Manager': 16,
 'Data Engineering Manager': 17,
 'Machine Learning Infrastructure Engineer': 18,
 'ML Engineer': 19,
 'AI Scientist': 20,
 'Computer Vision Engineer': 21,
 'Principal Data Scientist': 22,
 'Data Science Manager': 23,
 'Head of Data': 24,
 '3D Computer Vision Researcher': 25,
 'Data Analytics Engineer': 26,
 'Applied Data Scientist': 27,
 'Marketing Data Analyst': 28,
 'Cloud Data Engineer': 29,
 'Financial Data Analyst': 30,
 'Computer Vision Software Engineer': 31,
 'Director of Data Engineering': 32,
 'Data Science Engineer': 33,
 'Principal Data Engineer': 34,
 'Machine Learning Developer': 35,
 'Applied Machine Learning Scientist': 36,
 'Data Analytics Manager': 37,
 'Head of Data Science': 38,
 'Data Specialist': 39,
 'Data Architect': 40,
 'Finance Data Analyst': 41,
 'Principal Data Analyst': 42,
```

```
    'Big Data Architect': 43,
    'Staff Data Scientist': 44,
    'Analytics Engineer': 45,
    'ETL Developer': 46,
    'Head of Machine Learning': 47,
    'NLP Engineer': 48,
    'Lead Machine Learning Engineer': 49,
    'Data Analytics Lead': 50}
```

ahora vamos a reemplazar los valores en el df

In [69]:
```
df["index_job"]= df['job_title'].replace(dict_job)
df
```

```
C:\Users\darly\AppData\Local\Temp\ipykernel_21952\2832190827.py:1: FutureWarning:

Downcasting behavior in `replace` is deprecated and will be removed in a future version. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
```

Out[69]:

	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_residence	remo
0	2020	MI	FT	Data Scientist	70000	EUR	79833	DE	
1	2020	SE	FT	Machine Learning Scientist	260000	USD	260000	JP	
2	2020	SE	FT	Big Data Engineer	85000	GBP	109024	GB	
3	2020	MI	FT	Product Data Analyst	20000	USD	20000	HN	
4	2020	SE	FT	Machine Learning Engineer	150000	USD	150000	US	
...	...	...	...	...	...	...	...	...	
602	2022	SE	FT	Data Engineer	154000	USD	154000	US	
603	2022	SE	FT	Data Engineer	126000	USD	126000	US	
604	2022	SE	FT	Data Analyst	129000	USD	129000	US	
605	2022	SE	FT	Data Analyst	150000	USD	150000	US	
606	2022	MI	FT	AI Scientist	200000	USD	200000	IN	

607 rows × 14 columns

In [70]:
```
df
```

Out[70]:

	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_residence	remo
0	2020	MI	FT	Data Scientist	70000	EUR	79833	DE	
1	2020	SE	FT	Machine Learning Scientist	260000	USD	260000	JP	
2	2020	SE	FT	Big Data Engineer	85000	GBP	109024	GB	
3	2020	MI	FT	Product Data Analyst	20000	USD	20000	HN	
4	2020	SE	FT	Machine Learning Engineer	150000	USD	150000	US	
...	...	...	...	...	...	...	...	...	
602	2022	SE	FT	Data Engineer	154000	USD	154000	US	
603	2022	SE	FT	Data Engineer	126000	USD	126000	US	
604	2022	SE	FT	Data Analyst	129000	USD	129000	US	
605	2022	SE	FT	Data Analyst	150000	USD	150000	US	
606	2022	MI	FT	AI Scientist	200000	USD	200000	IN	

607 rows × 14 columns

obtener solo data numerica

In [71]:
```
df_analisis = df.select_dtypes(include=['int64', 'float64'])
```

In [72]:
```
df_analisis
```

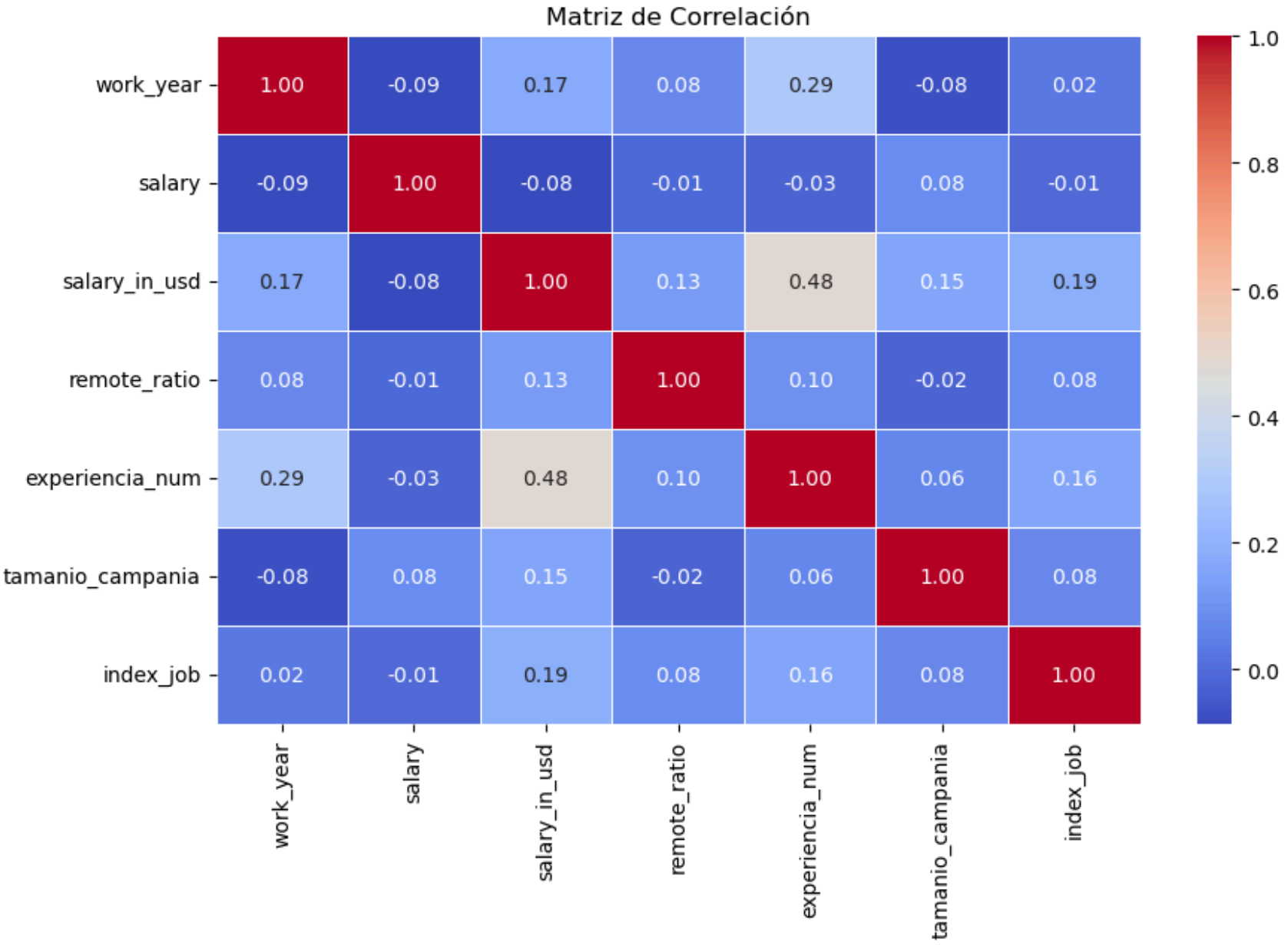
Out[72]:

	work_year	salary	salary_in_usd	remote_ratio	experiencia_num	tamanio_campania	index_job
0	2020	70000	79833	0	2	3	1
1	2020	260000	260000	0	3	1	2
2	2020	85000	109024	50	3	2	3
3	2020	20000	20000	0	2	1	4
4	2020	150000	150000	50	3	3	5
...	...	...	...	...	...	...	...
602	2022	154000	154000	100	3	2	11
603	2022	126000	126000	100	3	2	11
604	2022	129000	129000	0	3	2	6
605	2022	150000	150000	100	3	2	6
606	2022	200000	200000	100	2	3	20

607 rows × 7 columns

de nuevo, análisis de correclación

In [73]:
```
correlacion = df_analisis.corr()
#  🔹 Crear el mapa de calor
plt.figure(figsize=(10, 6))  # Ajustar tamaño de la figura
sns.heatmap(correlacion, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)

#  🔹 Título del gráfico
plt.title("Matriz de Correlación")

#  🔹 Mostrar el gráfico
plt.show()
```



NO cambio, toca aplicar oneHOtEncodeng

Guardar las diversas datas

In [71]:
```
df.to_csv(r"C:\Users\darly\OneDrive\Escritorio\materialClaseIA\dataSalarios\dataGeneral.csv", index=False)  # index=
df_analisis.to_csv(r"C:\Users\darly\OneDrive\Escritorio\materialClaseIA\dataSalarios\datosCuantitativos.csv", index=
```

Dividir data

Ya ahora nos vamos con la división de las datas para realizar los modelos de predicción y clasificación

In [74]:
```
#Librerias para modelos de machine learning
from sklearn.model_selection import train_test_split   #divide la data en entrenamiento y prueba
from sklearn.linear_model import LinearRegression      #apicar modelo de regresion lineal
from sklearn.metrics import mean_squared_error, r2_score # metricas del modelo
```

In [75]:
```
# Seleccionar la variable independiente (X) y la dependiente (y)
X = df[["experiencia_num"]] # Variable predictora, doble corchete para que retorno data frame y entre en el modelo
y = df['salary_in_usd']  # Variable objetivo da una serie
```

In [76]:
```
X
```

Out[76]:

	experiencia_num
0	2
1	3
2	3
3	2
4	3
...	...
602	3
603	3
604	3
605	3
606	2

607 rows × 1 columns

In [77]:
```
y
```

Out[77]:
```
0       79833
1      260000
2      109024
3       20000
4      150000
        ...
602    154000
603    126000
604    129000
605    150000
606    200000
Name: salary_in_usd, Length: 607, dtype: int64
```

Partir la data en sets de entranamiento y prueba 80% para entrenar el modelo 20% para evaluar su desempeño

In [78]:
```
# se usa para fijar la semilla del generador aleatorio, asegurando que los resultados sean reproducibles.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [79]:
```
X_train, X_test, y_train, y_test
```

Out[79]:
```
(     experiencia_num
 9                  3
 227                2
 591                3
 516                3
 132                2
 ..               ...
 71                 2
 106                2
 270                1
 435                2
 102                2

 [485 rows x 1 columns],
      experiencia_num
 563                3
 289                3
 76                 2
 78                 2
 182                2
 ..               ...
 249                3
 365                3
 453                2
 548                3
 235                2

 [122 rows x 1 columns],
 9      125000
 227     88654
 591    144854
 516    152500
 132     38400
         ...
 71      42197
 106    187442
 270     72500
 435     91614
 102     36259
 Name: salary_in_usd, Length: 485, dtype: int64,
 563    140250
 289    135000
```

```
76     100000
78     270000
182     26005
        ...
249    170000
365    138600
453    120000
548     99050
235    110000
Name: salary_in_usd, Length: 122, dtype: int64)
```

Aplicar un modelo

Crear y entrenar el modelo de regresión lineal

In [80]:
```
# Crear el modelo
modelo1 = LinearRegression()

# Entrenar el modelo con los datos de entrenamiento
modelo1.fit(X_train, y_train)
```

Out[80]:
```
▾ LinearRegression
LinearRegression()
```

Cuando entrenamos un modelo de Regresión Lineal con sklearn, el modelo encuentra una ecuación de la recta en la forma:

- $Y = mX + b$

Donde:

- m (pendiente) = Indica cuánto cambia el salario (Y) por cada unidad extra de experiencia (X).
- b (intersección o intercepto) = Es el salario estimado cuando la experiencia es 0.

In [81]:
```
# Obtener la pendiente (coeficiente) y la intersección con el eje Y
pendiente =    modelo1.coef_[0]
```

```
interseccion = modelo1.intercept_

print(f"Ecuación de la regresión: Salario = {pendiente:.2f} * Experiencia + {interseccion:.2f}") #2f se redolndea a d
```

```
Ecuación de la regresión: Salario = 44575.96 * Experiencia + 6897.15
```

Los siguientes son los resultados de la predicción

In [83]:
```
# Predecir los salarios en el conjunto de prueba, lo
y_pred_1 = modelo1.predict(X_test)
```

In [84]:
```
y_pred_1
```

Out[84]:
```
array([140625.04456583, 140625.04456583,  96049.08062684,  96049.08062684,
        96049.08062684,  96049.08062684,  51473.11668785,  51473.11668785,
        96049.08062684,  51473.11668785,  96049.08062684, 140625.04456583,
       140625.04456583,  96049.08062684, 140625.04456583, 140625.04456583,
        96049.08062684,  96049.08062684,  51473.11668785, 140625.04456583,
       140625.04456583,  96049.08062684, 140625.04456583,  96049.08062684,
        96049.08062684,  96049.08062684, 140625.04456583,  96049.08062684,
       140625.04456583,  96049.08062684,  51473.11668785, 140625.04456583,
       140625.04456583,  51473.11668785,  96049.08062684, 140625.04456583,
        96049.08062684, 140625.04456583, 140625.04456583, 140625.04456583,
       140625.04456583,  96049.08062684,  96049.08062684,  96049.08062684,
        96049.08062684, 140625.04456583, 140625.04456583,  96049.08062684,
        96049.08062684, 140625.04456583, 140625.04456583, 140625.04456583,
       140625.04456583, 185201.00850483,  51473.11668785, 140625.04456583,
        96049.08062684,  51473.11668785,  96049.08062684, 140625.04456583,
       140625.04456583,  96049.08062684, 140625.04456583,  96049.08062684,
       140625.04456583,  96049.08062684, 140625.04456583,  96049.08062684,
       140625.04456583,  51473.11668785, 140625.04456583, 140625.04456583,
       140625.04456583, 140625.04456583,  96049.08062684, 140625.04456583,
       140625.04456583,  51473.11668785,  96049.08062684,  96049.08062684,
       185201.00850483, 185201.00850483,  51473.11668785, 140625.04456583,
        96049.08062684, 140625.04456583,  96049.08062684,  96049.08062684,
        96049.08062684,  96049.08062684, 140625.04456583,  51473.11668785,
        51473.11668785,  96049.08062684,  96049.08062684, 140625.04456583,
        96049.08062684, 140625.04456583, 140625.04456583, 185201.00850483,
       140625.04456583,  51473.11668785, 140625.04456583, 140625.04456583,
       140625.04456583,  96049.08062684,  96049.08062684,  96049.08062684,
       140625.04456583, 140625.04456583,  51473.11668785,  96049.08062684,
        96049.08062684, 140625.04456583, 140625.04456583,  96049.08062684,
       140625.04456583,  96049.08062684])
```

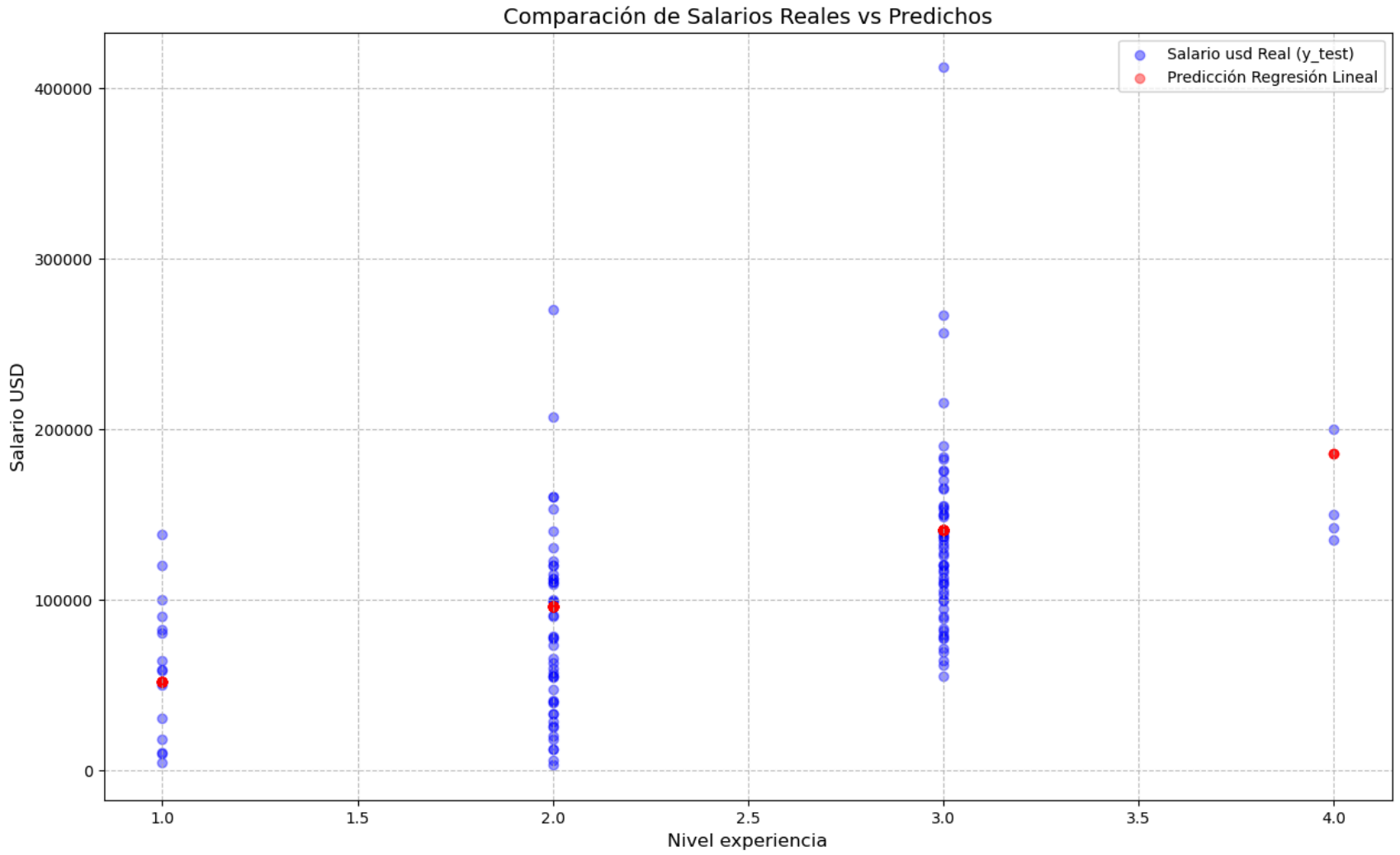
vamos a comparar los resultados

In [85]:
```
# Crear un DataFrame con los valores reales y las predicciones
resultados_predicciones = pd.DataFrame({
    "Salario en USD REAL (y_test)": y_test,
    "Predicción Regresión Lineal ": y_pred_1,

})
```

```
# Imprimir los primeros valores en formato de tabla
print(resultados_predicciones)  # Muestra solo las primeras filas
```

```
     Salario en USD REAL (y_test)  Predicción Regresión Lineal
563                        140250                140625.044566
289                        135000                140625.044566
76                         100000                 96049.080627
78                         270000                 96049.080627
182                         26005                 96049.080627
..                            ...                          ...
249                        170000                140625.044566
365                        138600                140625.044566
453                        120000                 96049.080627
548                         99050                140625.044566
235                        110000                 96049.080627

[122 rows x 2 columns]
```

In [86]:
```
# Crear gráfico de dispersión para comparar valores reales y predicciones
plt.figure(figsize=(15,9 ))

# Graficar valores reales (y_test) en azul
plt.scatter(X_test, y_test, color='blue', label="Salario usd Real (y_test)", alpha=0.4)# alpha puntos transperantes

# Graficar predicciones de Regresión Lineal en rojo
plt.scatter(X_test, y_pred_1, color='red', label="Predicción Regresión Lineal", alpha=0.4)

# Configurar el gráfico
plt.xlabel("Nivel experiencia", fontsize=12)
plt.ylabel("Salario USD", fontsize=12)
plt.title("Comparación de Salarios Reales vs Predichos", fontsize=14)
plt.legend()
plt.grid(True, linestyle="--", alpha=0.8)

# Mostrar gráfico
plt.show()
```



Calificar un modelo

- El Error Cuadrático Medio (MSE) mide cuánto se alejan las predicciones de los valores reales, calculando el promedio de los errores elevados al cuadrado. Como está en una escala diferente a los datos originales, se recomienda usar la Raíz del MSE (RMSE) para interpretarlo en la misma unidad de la variable objetivo. Un MSE bajo indica mayor precisión del modelo.

- Por otro lado, el Coeficiente de Determinación (R^2) mide qué porcentaje de la variabilidad de los datos es explicado por el modelo, con valores entre 0 y 1 (o negativos si el modelo es muy malo).
- Un R^2 cercano a 1 indica un buen ajuste, mientras que un valor bajo sugiere que el modelo no explica bien los datos.
- Para evaluar si el MSE es grande o pequeño, se debe comparar con la variabilidad de y_test, calculando su rango y desviación estándar.

In [87]:
```
# Calcular el error cuadrático medio (MSE)
mse = mean_squared_error(y_test, y_pred_1)

# Calcular el coeficiente de determinación R^2
r2 = r2_score(y_test, y_pred_1)

print(f"Error cuadrático medio (MSE): {mse:.2f}")
print(f"Coeficiente de determinación (R²): {r2:.2f}")
```

```
Error cuadrático medio (MSE): 3019758135.07
Coeficiente de determinación (R²): 0.21
```

Vamos a interpretar estos datos

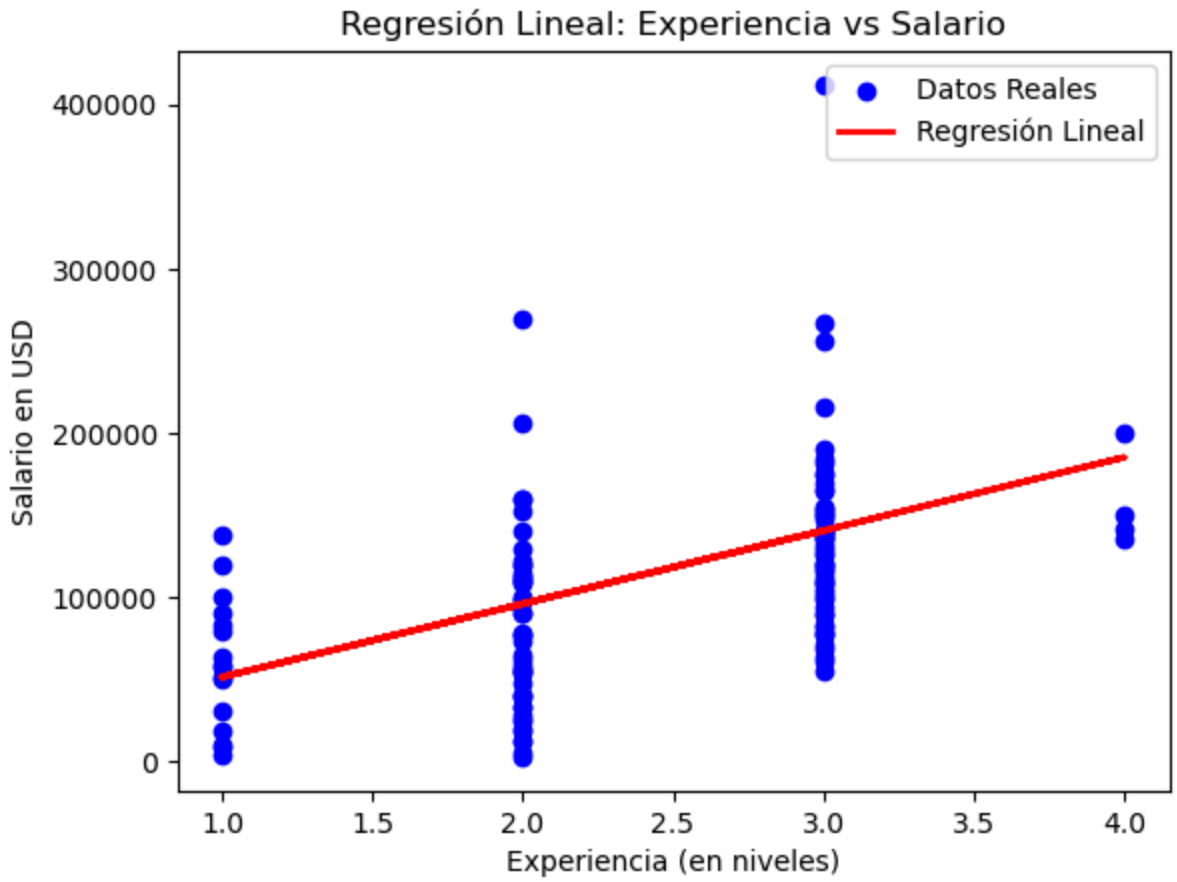
In [88]:
```
# Calcular el mínimo y máximo de y_test
min_y_test = y_test.min()
max_y_test = y_test.max()

# Calcular el rango de y_test
rango_y_test = max_y_test - min_y_test

# Calcular la desviación estándar de y_test
std_y_test= y_test.std()

#raiz cuadrada de MSE= RMSE
rmse=np.sqrt(mse)


# Mostrar los resultados

print(f"Rango de y_test: {rango_y_test:.2f}")
print(f"Desviación estándar de y_test: {std_y_test:.2f}")
print(f"Raíz de (MSE): (RMSE): {rmse:.2f}")
```

```
Rango de y_test: 409141.00
Desviación estándar de y_test: 62163.04
Raíz de (MSE): (RMSE): 54952.33
```

Interpretación

- El RMSE es menor que la desviación estándar → El modelo tiene cierto nivel de precisión.
- El R^2 es bajo (20%) → El modelo no explica bien los datos, lo que significa que:
- Hay otras variables importantes que no se incluyeron en el modelo.
- El modelo usado (Regresión Lineal) no es el mejor para este problema.
- Los datos pueden contener mucha aleatoriedad o ruido.

In [89]:
```
# Crear gráfico de dispersión
plt.scatter(X_test, y_test, color='blue', label="Datos Reales")
plt.plot(X_test, y_pred_1, color='red', linewidth=2, label="Regresión Lineal")

# Etiquetas
plt.xlabel("Experiencia (en niveles)")
plt.ylabel("Salario en USD")
plt.title("Regresión Lineal: Experiencia vs Salario")
plt.legend()

# Mostrar gráfico
plt.show()
```



reestructurar data

vamos a acomodar datos cualitativos a númericos.

one Hot encoding

In [90]:
```
#importar libreria
from sklearn.preprocessing import OneHotEncoder
```

se crea un objeto OneHotEncoder de la biblioteca sklearn.preprocessing, que se usa para convertir variables categóricas en una representación numérica de one-hot encoding.

Parámetros importantes:

- handle_unknown='ignore': Ignora las categorías desconocidas que puedan aparecer en los datos de prueba pero no en los datos de entrenamiento, evitando errores en la transformación.
- sparse=False: Retorna la matriz de salida como un array denso de NumPy en lugar de una matriz dispersa (por defecto es True, lo que produce una matriz dispersa).

In [91]:
```
encoder = OneHotEncoder(handle_unknown='ignore', sparse_output=False)
```

In [92]:
```
#transformar los datos  a oneHot
encoded_columns = encoder.fit_transform(df[['job_title']])
```

In [93]:
```
encoded_columns
```

Out[93]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.]])
```

In [94]:
```
#crear un dataFrame con la codificación
data_encode= pd.DataFrame(encoded_columns)
data_encode
```

Out[94]:

	0	1	2	3	4	5	6	7	8	9	...	40	41	42	43	44	45	46	47	48	49
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
602	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
603	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
604	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
605	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
606	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

607 rows × 50 columns

In [95]:
```
#convertir en un dataframe pero asignarle nombres a las columnas
data_encode= pd.DataFrame(encoded_columns,columns=encoder.get_feature_names_out(['job_title']))
```

In [96]:
```
data_encode
```

Out[96]:

	job_title_3D Computer Vision Researcher	job_title_AI Scientist	job_title_Analytics Engineer	job_title_Applied Data Scientist	job_title_Applied Machine Learning Scientist	job_title_BI Data Analyst	job_title_Big Data Architect	job_title_Big Data Engineer	job
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	
602	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
603	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
604	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
605	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
606	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	

607 rows × 50 columns

Revisemos como estan los datos desde el inicio

In [97]:
```
df_analisis
```

Out[97]:

	work_year	salary	salary_in_usd	remote_ratio	experiencia_num	tamanio_campania	index_job
0	2020	70000	79833	0	2	3	1
1	2020	260000	260000	0	3	1	2
2	2020	85000	109024	50	3	2	3
3	2020	20000	20000	0	2	1	4
4	2020	150000	150000	50	3	3	5
...	...	...	...	...	...	...	...
602	2022	154000	154000	100	3	2	11
603	2022	126000	126000	100	3	2	11
604	2022	129000	129000	0	3	2	6
605	2022	150000	150000	100	3	2	6
606	2022	200000	200000	100	2	3	20

607 rows × 7 columns

vamos a unir la data anterior con la nueva codificada

In [98]:
```
# Combinar datos procesados
df_final = pd.concat([data_encode,df_analisis],axis=1)
df_final
```

Out[98]:

	job_title_3D Computer Vision Researcher	job_title_AI Scientist	job_title_Analytics Engineer	job_title_Applied Data Scientist	job_title_Applied Machine Learning Scientist	job_title_BI Data Analyst	job_title_Big Data Architect	job_title_Big Data Engineer	job
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	
602	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
603	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
604	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
605	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
606	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	

607 rows × 57 columns

construir corr y su diagrama

In [99]:
```
cor=df_final.corr()
cor
```

Out[99]:

	job_title_3D Computer Vision Researcher	job_title_AI Scientist	job_title_Analytics Engineer	job_title_Applied Data Scientist	job_title_Applied Machine Learning Scientist	job_title_BI Data Analyst	job_title_Big Data Architect	jo
job_title_3D Computer Vision Researcher	1.000000	-0.004388	-0.003309	-0.003702	-0.003309	-0.004059	-0.001650	
job_title_AI Scientist	-0.004388	1.000000	-0.008797	-0.009844	-0.008797	-0.010792	-0.004388	
job_title_Analytics Engineer	-0.003309	-0.008797	1.000000	-0.007423	-0.006633	-0.008138	-0.003309	
job_title_Applied Data Scientist	-0.003702	-0.009844	-0.007423	1.000000	-0.007423	-0.009106	-0.003702	
job_title_Applied Machine Learning Scientist	-0.003309	-0.008797	-0.006633	-0.007423	1.000000	-0.008138	-0.003309	
job_title_BI Data Analyst	-0.004059	-0.010792	-0.008138	-0.009106	-0.008138	1.000000	-0.004059	
job_title_Big Data Architect	-0.001650	-0.004388	-0.003309	-0.003702	-0.003309	-0.004059	1.000000	
job_title_Big Data Engineer	-0.004695	-0.012483	-0.009412	-0.010532	-0.009412	-0.011547	-0.004695	
job_title_Business Data Analyst	-0.003702	-0.009844	-0.007423	-0.008306	-0.007423	-0.009106	-0.003702	
job_title_Cloud Data Engineer	-0.002336	-0.006210	-0.004683	-0.005240	-0.004683	-0.005745	-0.002336	
job_title_Computer Vision Engineer	-0.004059	-0.010792	-0.008138	-0.009106	-0.008138	-0.009983	-0.004059	
job_title_Computer Vision Software Engineer	-0.002863	-0.007612	-0.005740	-0.006423	-0.005740	-0.007042	-0.002863	

	job_title_3D Computer Vision Researcher	job_title_AI Scientist	job_title_Analytics Engineer	job_title_Applied Data Scientist	job_title_Applied Machine Learning Scientist	job_title_BI Data Analyst	job_title_Big Data Architect	jo
job_title_Data Analyst	-0.017716	-0.047106	-0.035520	-0.039745	-0.035520	-0.043575	-0.017716	
job_title_Data Analytics Engineer	-0.003309	-0.008797	-0.006633	-0.007423	-0.006633	-0.008138	-0.003309	
job_title_Data Analytics Lead	-0.001650	-0.004388	-0.003309	-0.003702	-0.003309	-0.004059	-0.001650	
job_title_Data Analytics Manager	-0.004388	-0.011667	-0.008797	-0.009844	-0.008797	-0.010792	-0.004388	
job_title_Data Architect	-0.005519	-0.014674	-0.011065	-0.012381	-0.011065	-0.013574	-0.005519	
job_title_Data Engineer	-0.021414	-0.056939	-0.042935	-0.048043	-0.042935	-0.052672	-0.021414	
job_title_Data Engineering Manager	-0.003702	-0.009844	-0.007423	-0.008306	-0.007423	-0.009106	-0.003702	
job_title_Data Science Consultant	-0.004388	-0.011667	-0.008797	-0.009844	-0.008797	-0.010792	-0.004388	
job_title_Data Science Engineer	-0.002863	-0.007612	-0.005740	-0.006423	-0.005740	-0.007042	-0.002863	
job_title_Data Science Manager	-0.005769	-0.015339	-0.011567	-0.012943	-0.011567	-0.014190	-0.005769	
job_title_Data Scientist	-0.022551	-0.059963	-0.045215	-0.050594	-0.045215	-0.055469	-0.022551	
job_title_Data Specialist	-0.001650	-0.004388	-0.003309	-0.003702	-0.003309	-0.004059	-0.001650	

	job_title_3D Computer Vision Researcher	job_title_AI Scientist	job_title_Analytics Engineer	job_title_Applied Data Scientist	job_title_Applied Machine Learning Scientist	job_title_BI Data Analyst	job_title_Big Data Architect	jo
job_title_Director of Data Engineering	-0.002336	-0.006210	-0.004683	-0.005240	-0.004683	-0.005745	-0.002336	
job_title_Director of Data Science	-0.004388	-0.011667	-0.008797	-0.009844	-0.008797	-0.010792	-0.004388	
job_title_ETL Developer	-0.002336	-0.006210	-0.004683	-0.005240	-0.004683	-0.005745	-0.002336	
job_title_Finance Data Analyst	-0.001650	-0.004388	-0.003309	-0.003702	-0.003309	-0.004059	-0.001650	
job_title_Financial Data Analyst	-0.002336	-0.006210	-0.004683	-0.005240	-0.004683	-0.005745	-0.002336	
job_title_Head of Data	-0.003702	-0.009844	-0.007423	-0.008306	-0.007423	-0.009106	-0.003702	
job_title_Head of Data Science	-0.003309	-0.008797	-0.006633	-0.007423	-0.006633	-0.008138	-0.003309	
job_title_Head of Machine Learning	-0.001650	-0.004388	-0.003309	-0.003702	-0.003309	-0.004059	-0.001650	
job_title_Lead Data Analyst	-0.002863	-0.007612	-0.005740	-0.006423	-0.005740	-0.007042	-0.002863	
job_title_Lead Data Engineer	-0.004059	-0.010792	-0.008138	-0.009106	-0.008138	-0.009983	-0.004059	
job_title_Lead Data Scientist	-0.002863	-0.007612	-0.005740	-0.006423	-0.005740	-0.007042	-0.002863	
job_title_Lead Machine Learning Engineer	-0.001650	-0.004388	-0.003309	-0.003702	-0.003309	-0.004059	-0.001650	

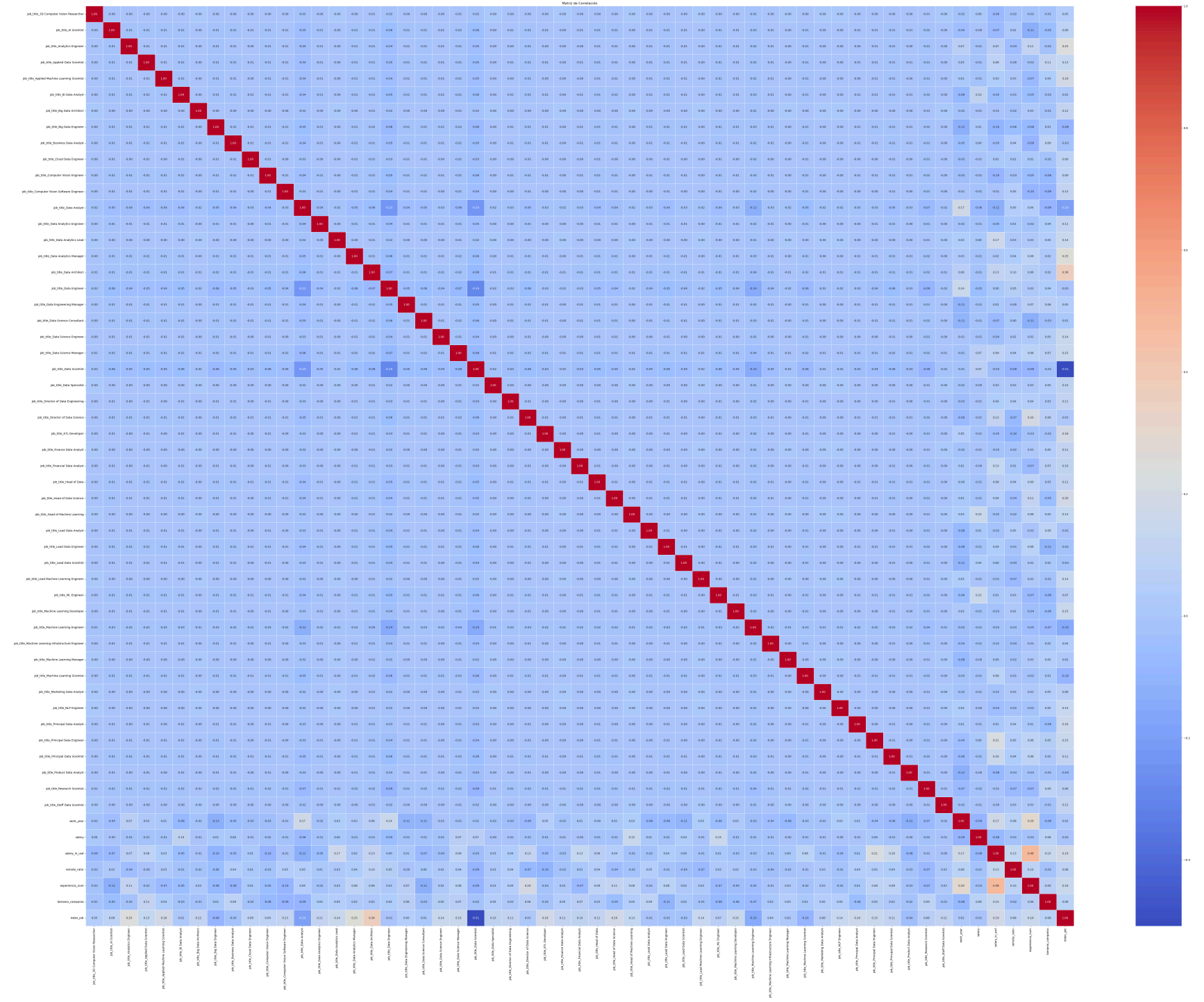
	job_title_3D Computer Vision Researcher	job_title_AI Scientist	job_title_Analytics Engineer	job_title_Applied Data Scientist	job_title_Applied Machine Learning Scientist	job_title_BI Data Analyst	job_title_Big Data Architect	jo
job_title_ML Engineer	-0.004059	-0.010792	-0.008138	-0.009106	-0.008138	-0.009983	-0.004059	
job_title_Machine Learning Developer	-0.002863	-0.007612	-0.005740	-0.006423	-0.005740	-0.007042	-0.002863	
job_title_Machine Learning Engineer	-0.010933	-0.029071	-0.021921	-0.024528	-0.021921	-0.026892	-0.010933	
job_title_Machine Learning Infrastructure Engineer	-0.002863	-0.007612	-0.005740	-0.006423	-0.005740	-0.007042	-0.002863	
job_title_Machine Learning Manager	-0.001650	-0.004388	-0.003309	-0.003702	-0.003309	-0.004059	-0.001650	
job_title_Machine Learning Scientist	-0.004695	-0.012483	-0.009412	-0.010532	-0.009412	-0.011547	-0.004695	
job_title_Marketing Data Analyst	-0.001650	-0.004388	-0.003309	-0.003702	-0.003309	-0.004059	-0.001650	
job_title_NLP Engineer	-0.001650	-0.004388	-0.003309	-0.003702	-0.003309	-0.004059	-0.001650	
job_title_Principal Data Analyst	-0.002336	-0.006210	-0.004683	-0.005240	-0.004683	-0.005745	-0.002336	
job_title_Principal Data Engineer	-0.002863	-0.007612	-0.005740	-0.006423	-0.005740	-0.007042	-0.002863	
job_title_Principal Data Scientist	-0.004388	-0.011667	-0.008797	-0.009844	-0.008797	-0.010792	-0.004388	
job_title_Product Data Analyst	-0.002336	-0.006210	-0.004683	-0.005240	-0.004683	-0.005745	-0.002336	

	job_title_3D Computer Vision Researcher	job_title_AI Scientist	job_title_Analytics Engineer	job_title_Applied Data Scientist	job_title_Applied Machine Learning Scientist	job_title_BI Data Analyst	job_title_Big Data Architect	jo
job_title_Research Scientist	-0.006684	-0.017772	-0.013401	-0.014995	-0.013401	-0.016440	-0.006684	
job_title_Staff Data Scientist	-0.001650	-0.004388	-0.003309	-0.003702	-0.003309	-0.004059	-0.001650	
work_year	-0.023806	-0.040985	0.070042	0.025662	0.011156	-0.082634	-0.023806	
salary	0.002001	-0.002340	-0.007864	-0.008954	-0.009641	0.102181	-0.005239	
salary_in_usd	-0.061243	-0.070327	0.072030	0.081441	0.034200	-0.052908	-0.007216	
remote_ratio	-0.020895	0.020311	-0.041894	-0.002067	0.033194	-0.010454	-0.020895	
experiencia_num	-0.020805	-0.114300	0.113943	-0.023453	-0.067657	-0.051174	0.030952	
tamanio_campania	-0.011777	-0.054927	-0.023613	0.113039	0.038704	-0.028968	-0.011777	
index_job	0.051112	0.086976	0.250058	0.131184	0.183647	0.017091	0.117359	

57 rows × 57 columns

In [100…

```
#  🔹 Crear el mapa de calor
plt.figure(figsize=(80, 60))  # Ajustar tamaño de la figura
sns.heatmap(cor, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)

#  🔹 Título del gráfico
plt.title("Matriz de Correlación")

#  🔹 Mostrar el gráfico
plt.show()
```


Matriz de Correlación

aplicar regresión lineal múltiple

Vamos a separar los datos para x y y, tener en cuenta que los datos de X son múltiples

In [101…
```
# Separar variables predictoras y objetivo
X = df_final.drop(columns=['salary_in_usd'])
y = df_final['salary_in_usd']
```

In [102…
```
# Dividir en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [103…
```
# elegir modelo
modelo2 = LinearRegression()
# Entrenar modelos
modelo2.fit(X_train, y_train)
```

Out[103…
```
▾ LinearRegression
LinearRegression()
```

In [106…
```
# Hacer predicciones
y_pred_2 = modelo2.predict(X_test)
```

In [107…
```
# Evaluar modelos
mse = mean_squared_error(y_test, y_pred_2)
r2 = r2_score(y_test, y_pred_2)
```

In [108…
```
# Calcular el mínimo y máximo de y_test
min_y_test = y_test.min()
max_y_test = y_test.max()

# Calcular el rango de y_test
rango_y_test = max_y_test - min_y_test

# Calcular la desviación estándar de y_test
std_y_test= y_test.std()
```

```
#raiz cuadrada de MSE= RMSE
rmse=np.sqrt(mse)

# Mostrar los resultados

print(f"Rango de y_test: {rango_y_test:.2f}")
print(f"Desviación estándar de y_test: {std_y_test:.2f}")
print(f"Raíz de (MSE): (RMSE): {rmse:.2f}")
print(f"Coeficiente de determinación (R²): {r2:.2f}")
```

```
Rango de y_test: 409141.00
Desviación estándar de y_test: 62163.04
Raíz de (MSE): (RMSE): 61377.97
Coeficiente de determinación (R²): 0.02
```

Nos hemos dado cuenta que el modelo desmejoró casi en su totalidad, probemos con nuevas características

Probar otras características

In [109…

```
#crear objeto de encoder, transformar los datos y pasarlos al dataFrame
encoder = OneHotEncoder(handle_unknown='ignore', sparse_output=False)
encoded_columns = encoder.fit_transform(df[['company_location']])
data_encode2= pd.DataFrame(encoded_columns,columns=encoder.get_feature_names_out(['company_location']))
data_encode2
```

Out[109…

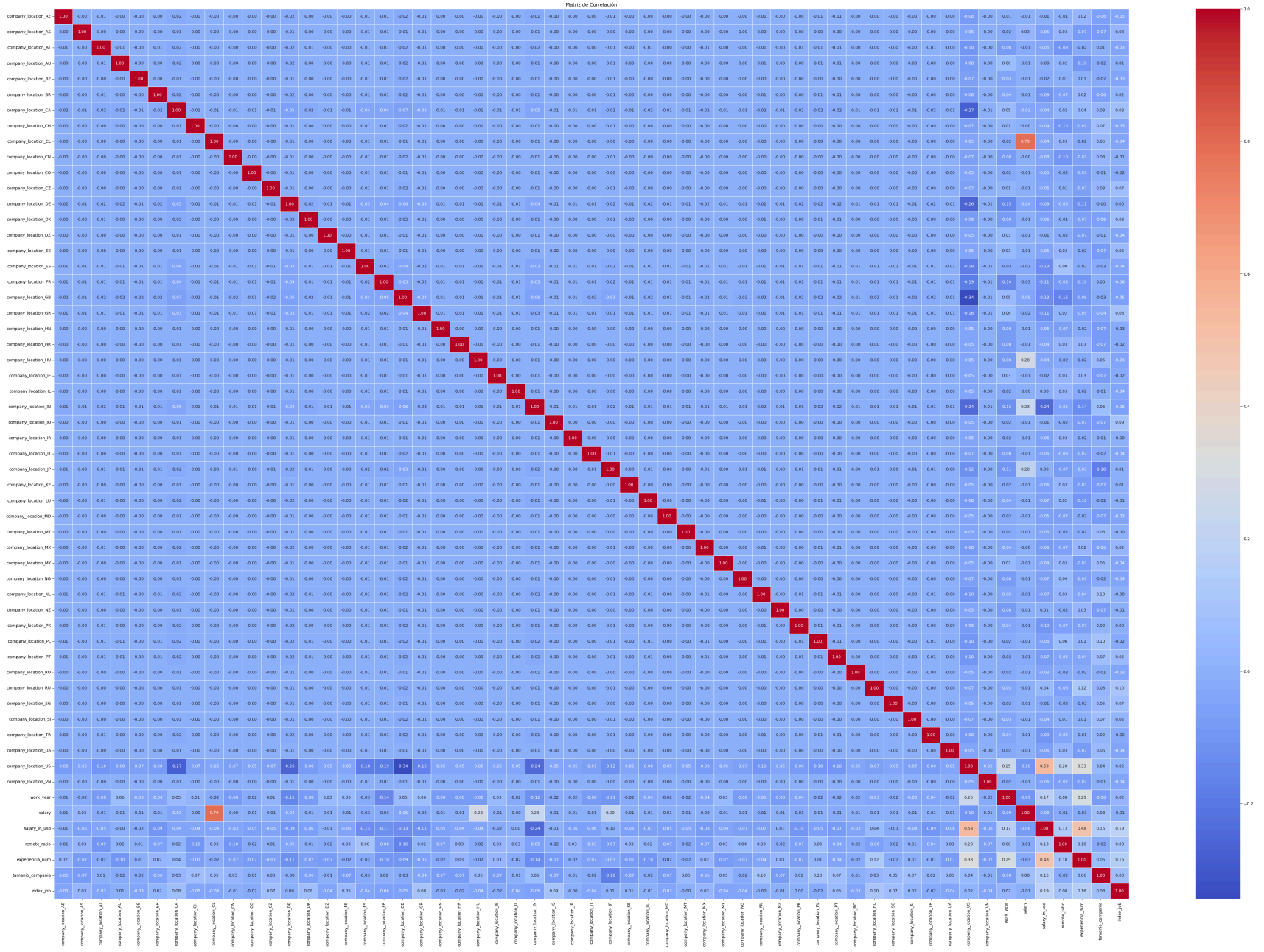
	company_location_AE	company_location_AS	company_location_AT	company_location_AU	company_location_BE	company_lo
0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	
602	0.0	0.0	0.0	0.0	0.0	
603	0.0	0.0	0.0	0.0	0.0	
604	0.0	0.0	0.0	0.0	0.0	
605	0.0	0.0	0.0	0.0	0.0	
606	0.0	0.0	0.0	0.0	0.0	

607 rows × 50 columns

In [110…
```
#unir las datas de analisis y la de codificación
df_final = pd.concat([data_encode2,df_analisis],axis=1)
```

In [111…
```
#obtener correlación y gráficarlo
cor=df_final.corr()
```

In [112…
```
#  🔹 Crear el mapa de calor
plt.figure(figsize=(60, 40))  # Ajustar tamaño de la figura
sns.heatmap(cor, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)

#  🔹 Título del gráfico
plt.title("Matriz de Correlación")

#  🔹 Mostrar el gráfico
plt.show()
```



In [113…

```
# Separar variables predictoras y objetivo
X = df_final.drop(columns=['salary_in_usd'])
```

```
y = df_final['salary_in_usd']
df_final
```

Out[113…

	company_location_AE	company_location_AS	company_location_AT	company_location_AU	company_location_BE	company_lo
0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	
602	0.0	0.0	0.0	0.0	0.0	
603	0.0	0.0	0.0	0.0	0.0	
604	0.0	0.0	0.0	0.0	0.0	
605	0.0	0.0	0.0	0.0	0.0	
606	0.0	0.0	0.0	0.0	0.0	

607 rows × 57 columns

In [120…

```
# Dividir en entrenamiento y prueba
X_train_3, X_test_3, y_train_3, y_test_3 = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [121…

```
#elegir modelo
modelo3 = LinearRegression()
# Entrenar modelos
modelo3.fit(X_train_3, y_train_3)
```

Out[121…

▾ LinearRegression

```
LinearRegression()
```

In [122…
```
# Hacer predicciones
y_pred_3 = modelo3.predict(X_test_3)
```

In [123…
```
# Evaluar modelos
mse_3 = mean_squared_error(y_test_3, y_pred_3)
r2_3 = r2_score(y_test_3, y_pred_3)
```

In [124…
```
# Calcular el mínimo y máximo de y_test
min_y_test_3 = y_test_3.min()
max_y_test_3 = y_test_3.max()

# Calcular el rango de y_test
rango_y_test_3 = max_y_test_3 - min_y_test_3

# Calcular la desviación estándar de y_test
std_y_test_3= y_test_3.std()

#raiz cuadrada de MSE= RMSE
rmse_3=np.sqrt(mse_3)

# Mostrar los resultados

print(f"Rango de y_test: {rango_y_test_3:.2f}")
print(f"Desviación estándar de y_test: {std_y_test_3:.2f}")
print(f"Raíz de (MSE): (RMSE): {rmse_3:.2f}")
print(f"Coeficiente de determinación (R²): {r2_3:.2f}")
```

```
Rango de y_test: 409141.00
Desviación estándar de y_test: 62163.04
Raíz de (MSE): (RMSE): 48970.17
Coeficiente de determinación (R²): 0.37
```

este modelo mejoro bastante en todas las medidas, hasta ahora es el mejor que tenemos

Gráficar y_pred y y_test

In [125…
```
# Crear un DataFrame con los valores reales y las predicciones
resultados_predicciones = pd.DataFrame({
    "Salario en USD REAL (y_test_3)": y_test_3,
    "Predicción Regresión Lineal model 3 ": y_pred_3,
```

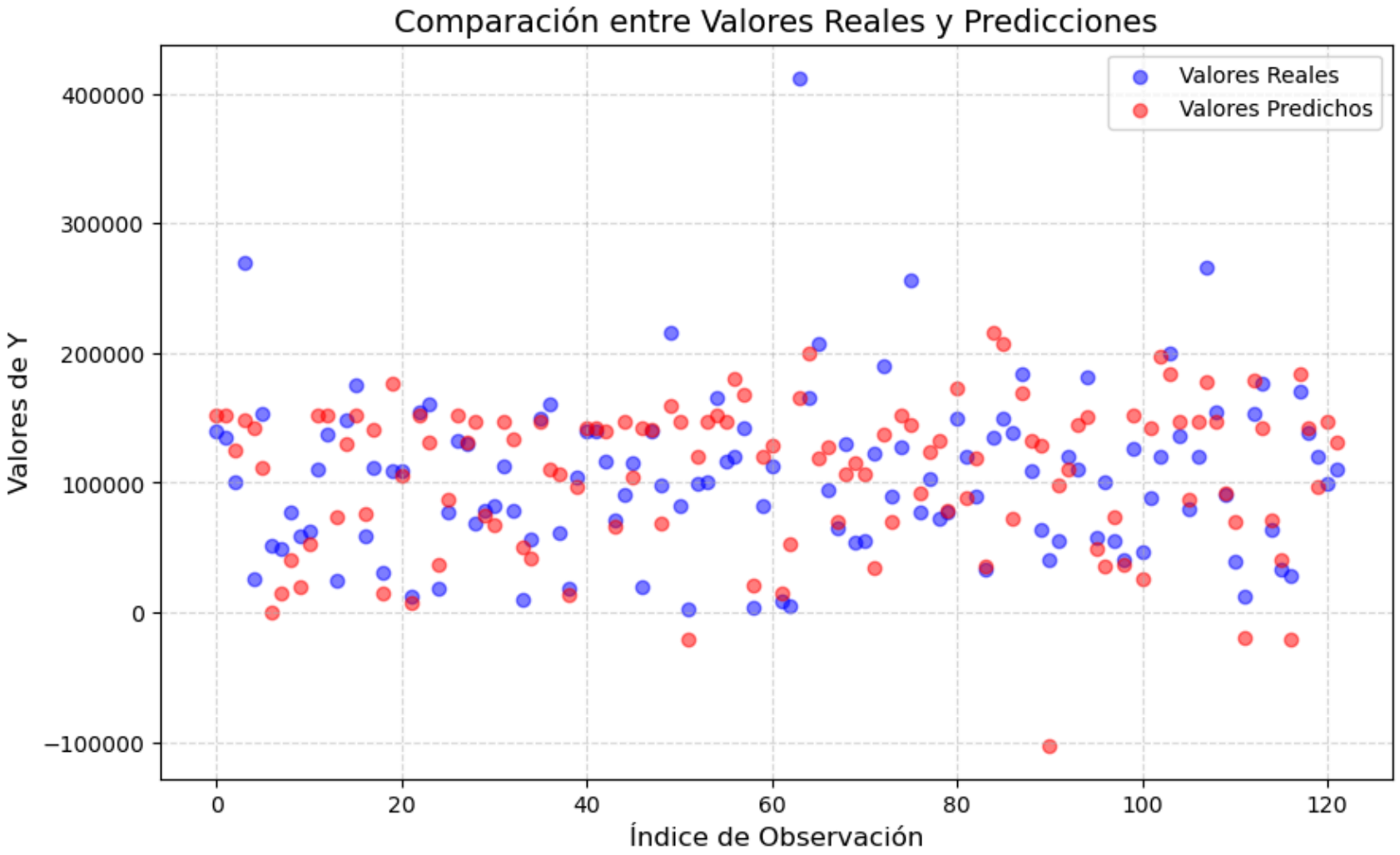
```
})

# Imprimir los primeros valores en formato de tabla
print(resultados_predicciones)  # Muestra solo las primeras filas
```

```
     Salario en USD REAL (y_test_3)  Predicción Regresión Lineal model 3
563                          140250                         151792.695731
289                          135000                         151822.531968
76                           100000                         125616.675281
78                           270000                         147996.035022
182                           26005                         141707.886136
..                              ...                                   ...
249                          170000                         184347.347725
365                          138600                         142403.356931
453                          120000                          96690.080029
548                           99050                         147327.481202
235                          110000                         131171.794970

[122 rows x 2 columns]
```

mirar a traves de un scatter

In [127…

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))

# Graficar cada punto (valor real vs predicho)
plt.scatter(range(len(y_test_3)), y_test_3, color='blue', label="Valores Reales", alpha=0.5)
plt.scatter(range(len(y_pred_3)), y_pred_3, color='red', label="Valores Predichos", alpha=0.5)

# Configurar el gráfico
plt.xlabel("Índice de Observación", fontsize=12)
plt.ylabel("Valores de Y", fontsize=12)
plt.title("Comparación entre Valores Reales y Predicciones", fontsize=14)
plt.legend()
plt.grid(True, linestyle="--", alpha=0.5)

# Mostrar gráfico
plt.show()
```



Guardar modelos

instalar libreria que permite guardar los modelos

In [128…

```
conda install joblib
```

```
Retrieving notices: done
Channels:
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): ...working... done
Solving environment: ...working... done

# All requested packages already installed.


Note: you may need to restart the kernel to use updated packages.
```

In [130…
```
# importar librería
import joblib
```

Vamos a guardar los 3 modelos y luego la mejor data

In [132…
```
# Guardar el modelo 1
joblib.dump(modelo1, 'modelo_entrenado_columnaExperienciaSC.pkl')
```

Out[132… ['modelo_entrenado_columnaExperienciaSC.pkl']

In [133…
```
# Guardar el modelo 3, porque el 2 no merece
joblib.dump(modelo3, 'modelo3_RegMUl_encodeCompany.pkl')
```

Out[133… ['modelo3_RegMUl_encodeCompany.pkl']

Transformar el resto de datos, unirlos a la data de entrenamiento y probar modelo

In [137…
```
#unir las data final (codificac¿da) y la codificación de job_tittle
df_final_1 = pd.concat([df_final,data_encode],axis=1)
```

In [138…
```
df_final_1
```

Out[138…

	company_location_AE	company_location_AS	company_location_AT	company_location_AU	company_location_BE	company_lo
0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	
602	0.0	0.0	0.0	0.0	0.0	
603	0.0	0.0	0.0	0.0	0.0	
604	0.0	0.0	0.0	0.0	0.0	
605	0.0	0.0	0.0	0.0	0.0	
606	0.0	0.0	0.0	0.0	0.0	

607 rows × 107 columns

In [140…

```
for i in df_final_1.columns:
    print(i)
```

```
company_location_AE
company_location_AS
company_location_AT
company_location_AU
company_location_BE
company_location_BR
company_location_CA
company_location_CH
company_location_CL
company_location_CN
company_location_CO
company_location_CZ
company_location_DE
company_location_DK
company_location_DZ
company_location_EE
company_location_ES
company_location_FR
company_location_GB
company_location_GR
company_location_HN
company_location_HR
company_location_HU
company_location_IE
company_location_IL
company_location_IN
company_location_IQ
company_location_IR
company_location_IT
company_location_JP
company_location_KE
company_location_LU
company_location_MD
company_location_MT
company_location_MX
company_location_MY
company_location_NG
company_location_NL
company_location_NZ
company_location_PK
company_location_PL
company_location_PT
```

```
company_location_RO
company_location_RU
company_location_SG
company_location_SI
company_location_TR
company_location_UA
company_location_US
company_location_VN
work_year
salary
salary_in_usd
remote_ratio
experiencia_num
tamanio_campania
index_job
job_title_3D Computer Vision Researcher
job_title_AI Scientist
job_title_Analytics Engineer
job_title_Applied Data Scientist
job_title_Applied Machine Learning Scientist
job_title_BI Data Analyst
job_title_Big Data Architect
job_title_Big Data Engineer
job_title_Business Data Analyst
job_title_Cloud Data Engineer
job_title_Computer Vision Engineer
job_title_Computer Vision Software Engineer
job_title_Data Analyst
job_title_Data Analytics Engineer
job_title_Data Analytics Lead
job_title_Data Analytics Manager
job_title_Data Architect
job_title_Data Engineer
job_title_Data Engineering Manager
job_title_Data Science Consultant
job_title_Data Science Engineer
job_title_Data Science Manager
job_title_Data Scientist
job_title_Data Specialist
job_title_Director of Data Engineering
job_title_Director of Data Science
job_title_ETL Developer
```

```
job_title_Finance Data Analyst
job_title_Financial Data Analyst
job_title_Head of Data
job_title_Head of Data Science
job_title_Head of Machine Learning
job_title_Lead Data Analyst
job_title_Lead Data Engineer
job_title_Lead Data Scientist
job_title_Lead Machine Learning Engineer
job_title_ML Engineer
job_title_Machine Learning Developer
job_title_Machine Learning Engineer
job_title_Machine Learning Infrastructure Engineer
job_title_Machine Learning Manager
job_title_Machine Learning Scientist
job_title_Marketing Data Analyst
job_title_NLP Engineer
job_title_Principal Data Analyst
job_title_Principal Data Engineer
job_title_Principal Data Scientist
job_title_Product Data Analyst
job_title_Research Scientist
job_title_Staff Data Scientist
```

In [143…
```
df.columns #colu de la data oficial
```

Out[143…
```
Index(['work_year', 'experience_level', 'employment_type', 'job_title',
       'salary', 'salary_currency', 'salary_in_usd', 'employee_residence',
       'remote_ratio', 'company_location', 'company_size', 'experiencia_num',
       'tamanio_campania', 'index_job'],
      dtype='object')
```

In [ ]:
```
df['employment_type_numeric'] = df['employment_type'].map({'FL': 1, 'CT': 2, 'PT': 3, 'FT': 4})
df['remote_ratio_numeric'] = df['remote_ratio'] / 100
```

In [144…
```
#label encode para tipo empleo
df_final_1["tipo_empleo"]= df['employment_type'].map({'FL': 1, 'CT': 2, 'PT': 3, 'FT': 4})
```

In [145…
```
df_final_1.shape
```

Out[145…
```
(607, 108)
```

In [147…
```
#transformar datos en intervalor de 0 a 1,
df_final_1['remote_ratio']
```

Out[147…
```
0        0
1        0
2       50
3        0
4       50
      ...
602    100
603    100
604      0
605    100
606    100
Name: remote_ratio, Length: 607, dtype: int64
```

In [148…
```
df_final_1['remote_ratio_numeric'] = df['remote_ratio'] / 100
```

In [149…
```
df_final_1.shape
```

Out[149…
```
(607, 109)
```

eliminar las columnas que ya fueron codificadas

In [150…
```
df_final_1.drop(columns=['remote_ratio', 'index_job'], inplace=True)
```

In [152…
```
df_final_1.shape
```

Out[152…
```
(607, 107)
```

probar el modelo de regresión lineal multiple

In [158…
```
# Separar variables predictoras y objetivo
X = df_final_1.drop(columns=['salary_in_usd'])
y = df_final_1['salary_in_usd']
```

In [159…
```
# Dividir en entrenamiento y prueba
X_train_4, X_test_4, y_train_4, y_test_4 = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [160…
```
#elegir modelo
modelo4 = LinearRegression()
# Entrenar modelos
modelo4.fit(X_train_4, y_train_4)
```

Out[160…
```
LinearRegression()
```

In [161…
```
# Hacer predicciones
y_pred_4 = modelo4.predict(X_test_4)
```

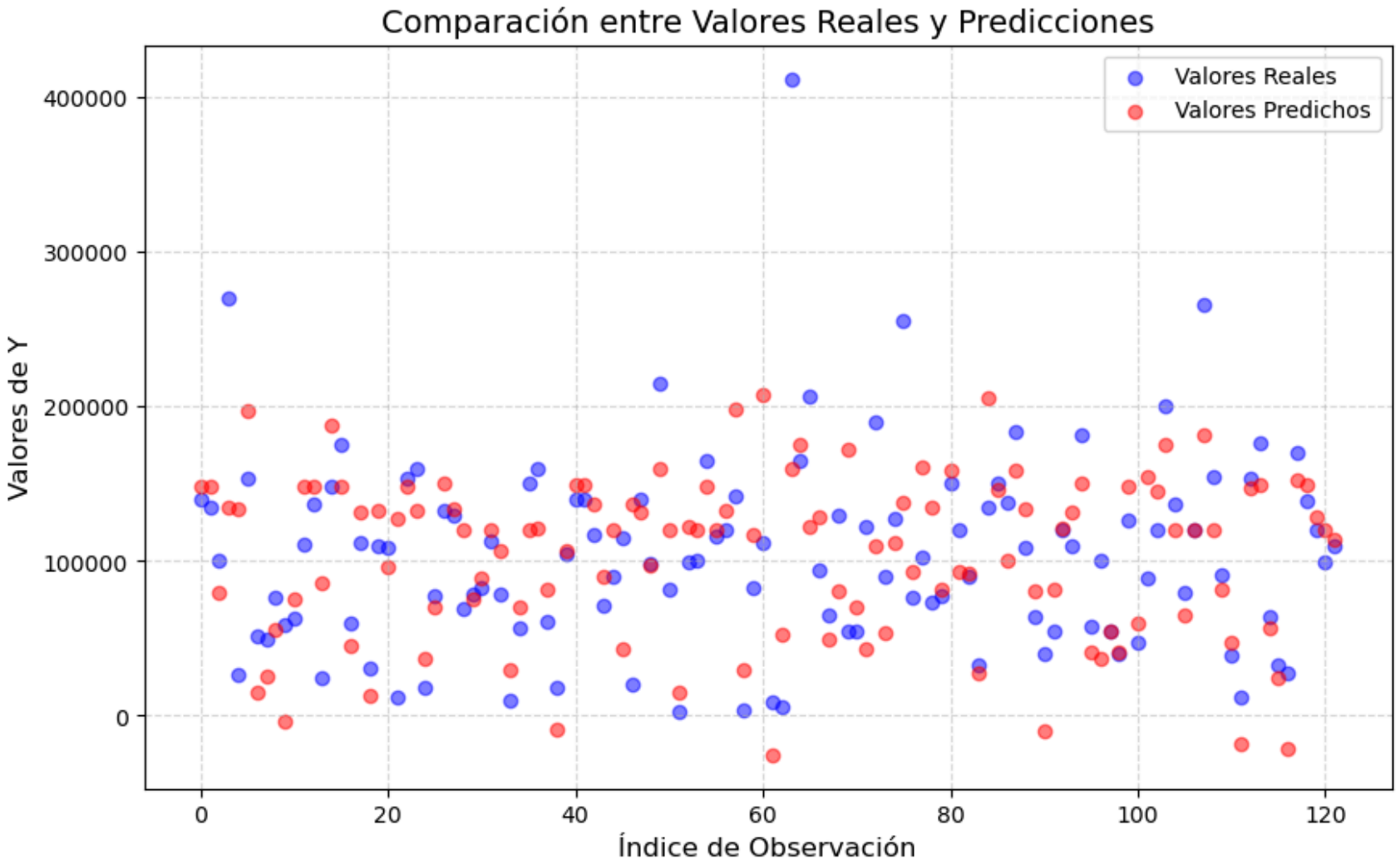
In [162…
```
# Evaluar modelos
mse_4 = mean_squared_error(y_test_4, y_pred_4)
r2_4 = r2_score(y_test_4, y_pred_4)
```

In [163…
```
# Calcular el mínimo y máximo de y_test
min_y_test_4 = y_test_4.min()
max_y_test_4 = y_test_4.max()

# Calcular el rango de y_test
rango_y_test_4 = max_y_test_4 - min_y_test_4

# Calcular la desviación estándar de y_test
std_y_test_4= y_test_4.std()

#raiz cuadrada de MSE= RMSE
rmse_4=np.sqrt(mse_4)

# Mostrar los resultados

print(f"Rango de y_test: {rango_y_test_4:.2f}")
print(f"Desviación estándar de y_test: {std_y_test_4:.2f}")
print(f"Raíz de (MSE): (RMSE): {rmse_4:.2f}")
print(f"Coeficiente de determinación (R²): {r2_4:.2f}")
```

```
Rango de y_test: 409141.00
Desviación estándar de y_test: 62163.04
Raíz de (MSE): (RMSE): 47710.45
Coeficiente de determinación (R²): 0.41
```

In [175…

```python
#mirar puntos
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))

# Graficar cada punto (valor real vs predicho)
plt.scatter(range(len(y_test_4)), y_test_4, color='blue', label="Valores Reales", alpha=0.5)
plt.scatter(range(len(y_pred_4)), y_pred_4, color='red', label="Valores Predichos", alpha=0.5)

# Configurar el gráfico
plt.xlabel("Índice de Observación", fontsize=12)
plt.ylabel("Valores de Y", fontsize=12)
plt.title("Comparación entre Valores Reales y Predicciones", fontsize=14)
plt.legend()
plt.grid(True, linestyle="--", alpha=0.5)

# Mostrar gráfico
plt.show()
```



In [165…
```
#guardar modelo 4

joblib.dump(modelo4, 'RMultiple_datTrasns.pkl')
```

Out[165…
```
['RMultiple_datTrasns.pkl']
```

In [166…
```
#guardadarData
df_final_1.to_csv("dataSalarios_final.csv", index=False)  # index=False evita guardar los índices
```

eliminar valores atipicos

hasta ahora este ha sido el mejor modelo, pero al revisar el gráfico nos damos cuenta que aparecen algunos puntos por fuera de rango, los vamos a transformar

- El rango intercuartílico es una medida estadística de dispersión que describe el rango en el que se encuentra el 50% central de los datos. Se calcula como la diferencia entre el tercer cuartil (Q3) y el primer cuartil (Q1), es decir, IQR = Q3 - Q1. Este rango es útil para identificar y eliminar valores atípicos (outliers), ya que los datos que se encuentran muy lejos del centro de la distribución pueden distorsionar los análisis. En este caso, se usa el IQR para filtrar los outliers en la columna salary_in_usd.
- La primera línea del código, Q1, Q3 = df_final_1['salary_in_usd'].quantile([0.25, 0.75]), calcula el primer y tercer cuartil de la columna de salarios en dólares. El cuartil 1 (Q1) representa el valor por debajo del cual se encuentra el 25% de los datos, mientras que el cuartil 3 (Q3) representa el valor por debajo del cual se encuentra el 75% de los datos.
- En la segunda línea, IQR = Q3 - Q1, se calcula el rango intercuartílico como la diferencia entre Q3 y Q1, lo cual representa el rango que contiene el 50% de los datos centrales.
- La tercera línea, df_final_1 = df_final_1[(df_final_1['salary_in_usd'] >= (Q1 - 1.5 * IQR)) & (df_final_1['salary_in_usd'] <= (Q3 + 1.5 * IQR))], filtra el DataFrame original, conservando únicamente las filas en las que el salario se encuentra dentro del rango permitido. Este rango se define entre Q1 menos 1.5 veces el IQR y Q3 más 1.5 veces el IQR. Los valores fuera de este rango se consideran outliers y se eliminan del conjunto de datos.
- Este proceso permite limpiar los datos eliminando valores extremos que podrían afectar negativamente el análisis estadístico o el entrenamiento de modelos predictivos.

In [167…
```
# Filtrar outliers en salario usando IQR
Q1, Q3 = df_final_1['salary_in_usd'].quantile([0.25, 0.75])
IQR = Q3 - Q1
df_final_1 = df_final_1[(df_final_1['salary_in_usd'] >= (Q1 - 1.5 * IQR)) & (df_final_1['salary_in_usd'] <= (Q3 + 1.5
```

In [168…
```
df_final_1.shape
```

Out[168… (597, 107)

probar modelo de nuevo

In [169…
```
# Separar variables predictoras y objetivo
X = df_final_1.drop(columns=['salary_in_usd'])
y = df_final_1['salary_in_usd']
```

In [170…
```
# Dividir en entrenamiento y prueba
X_train_5, X_test_5, y_train_5, y_test_5 = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [171…
```
#elegir modelo
modelo5 = LinearRegression()
# Entrenar modelos
modelo5.fit(X_train_5, y_train_5)
```

Out[171…
```
LinearRegression()
```

In [172…
```
# Hacer predicciones
y_pred_5 = modelo5.predict(X_test_5)
```

In [173…
```
# Evaluar modelos
mse_5 = mean_squared_error(y_test_5, y_pred_5)
r2_5 = r2_score(y_test_5, y_pred_5)
```

In [174…
```
# Calcular el mínimo y máximo de y_test
min_y_test_5 = y_test_5.min()
max_y_test_5 = y_test_5.max()

# Calcular el rango de y_test
rango_y_test_5 = max_y_test_5 - min_y_test_5

# Calcular la desviación estándar de y_test
std_y_test_5= y_test_5.std()

#raiz cuadrada de MSE= RMSE
rmse_5=np.sqrt(mse_5)
```

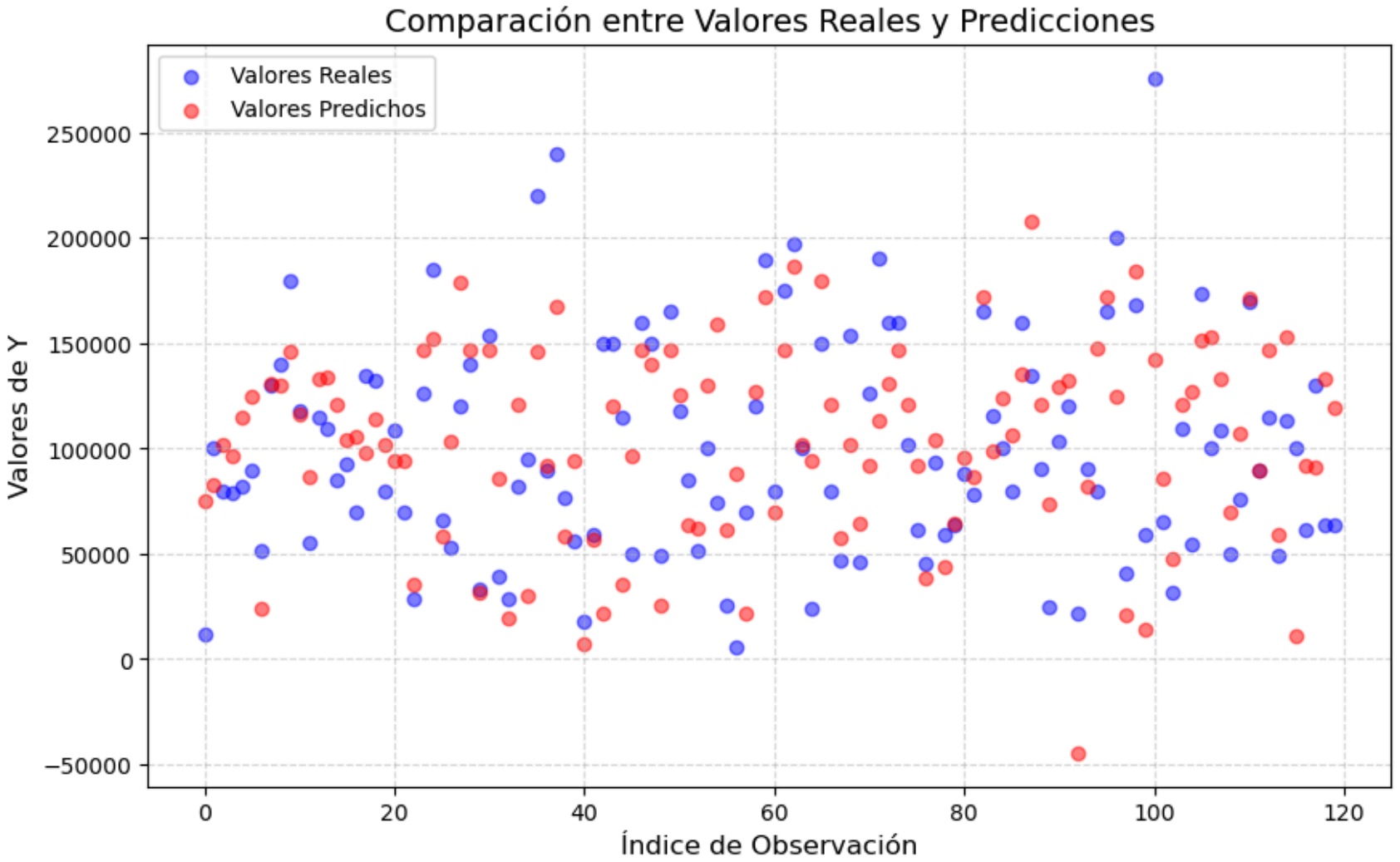
```
# Mostrar los resultados

print(f"Rango de y_test: {rango_y_test_5:.2f}")
print(f"Desviación estándar de y_test: {std_y_test_5:.2f}")
print(f"Raíz de (MSE): (RMSE): {rmse_5:.2f}")
print(f"Coeficiente de determinación (R²): {r2_5:.2f}")
```

```
Rango de y_test: 270118.00
Desviación estándar de y_test: 51826.73
Raíz de (MSE): (RMSE): 40044.95
Coeficiente de determinación (R²): 0.40
```

el modelo se ajusto un poco mejor

In [183…

```
#mirar puntos
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))

# Graficar cada punto (valor real vs predicho)
plt.scatter(range(len(y_test_5)), y_test_5, color='blue', label="Valores Reales", alpha=0.5)
plt.scatter(range(len(y_pred_5)), y_pred_5, color='red', label="Valores Predichos", alpha=0.5)



# Configurar el gráfico
plt.xlabel("Índice de Observación", fontsize=12)
plt.ylabel("Valores de Y", fontsize=12)
plt.title("Comparación entre Valores Reales y Predicciones", fontsize=14)
plt.legend()
plt.grid(True, linestyle="--", alpha=0.5)

# Mostrar gráfico
plt.show()
```



In [184…
```
#guardar modelo 5

joblib.dump(modelo5, 'RMultiple_outliner.pkl')
```

Out[184…
```
['RMultiple_outliner.pkl']
```

In [185…
```
#guardar la Data
df_final_1.to_csv("dataSalarios_final_outliner.csv", index=False)  # index=False evita guardar los índices
```

Arboles de decisión

In [ ]:
```
Vamos a probar arboles de decision con las dos datas
```

Random forest

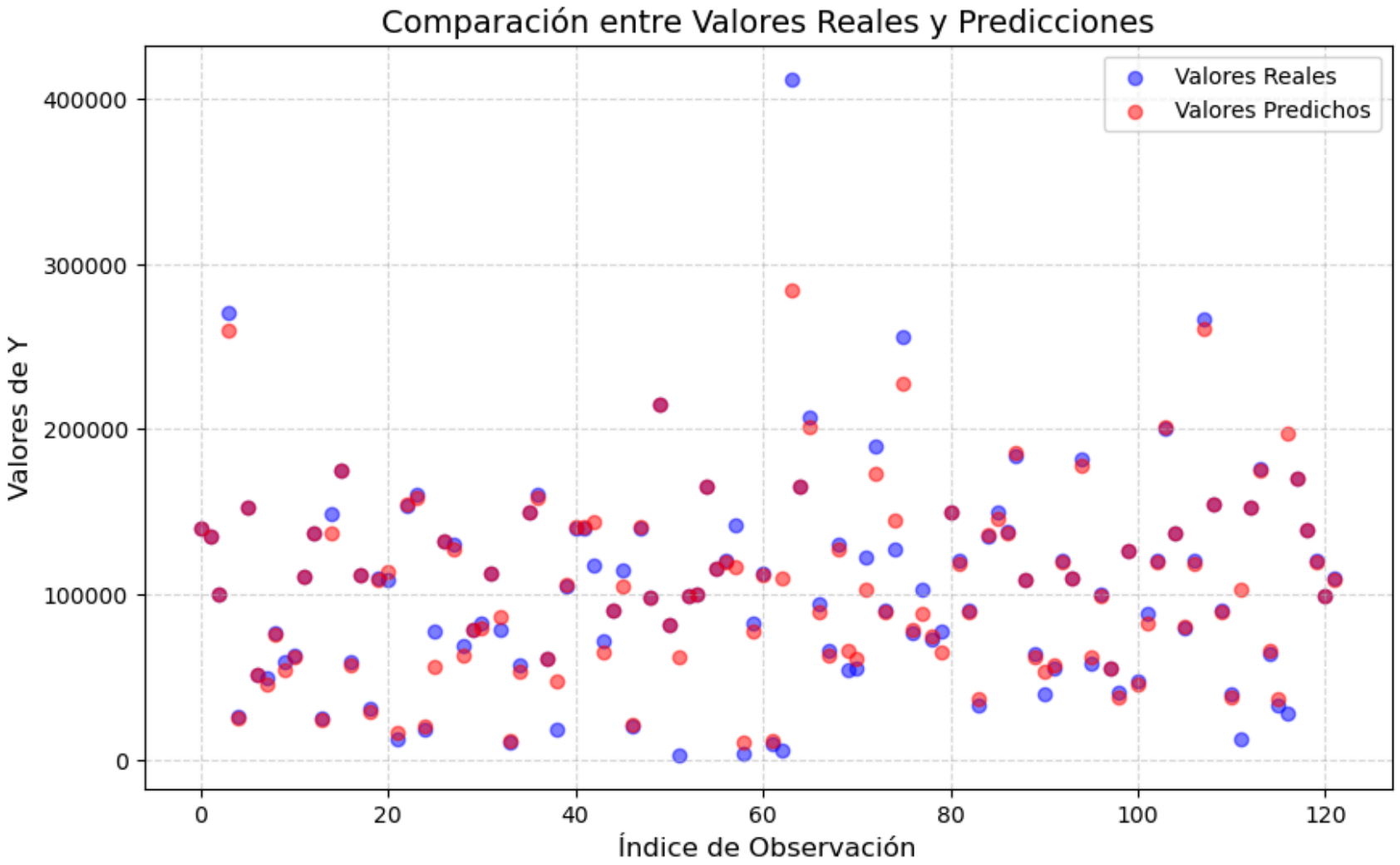
In [188…
```
# Entrenar modelos de randoms forest

from sklearn.ensemble import RandomForestRegressor

modelo_rf1 = RandomForestRegressor(n_estimators=100, random_state=42) #la data final con outliner
modelo_rf2 = RandomForestRegressor(n_estimators=100, random_state=42) #la data final sin outliner

modelo_rf1.fit(X_train_4, y_train_4)
modelo_rf2.fit(X_train_5, y_train_5)

# Hacer predicciones
y_pred_rf1 = modelo_rf1.predict(X_test_4)
y_pred_rf2 = modelo_rf2.predict(X_test_5)

# Evaluar modelos
mse_rf1 = mean_squared_error(y_test_4, y_pred_rf1)
r2_rf1 = r2_score(y_test_4, y_pred_rf1)
mse_rf2 = mean_squared_error(y_test_5, y_pred_rf2)
r2_rf2 = r2_score(y_test_5, y_pred_rf2)

# Comparación de modelos
comparacion_modelos = pd.DataFrame({
    "Modelo": ["Random forest 1", "Random Forest 2"],
    "MSE": [mse_rf1, mse_rf2],
    "R²": [r2_rf1, r2_rf2]
})

# Mostrar comparación
print(comparacion_modelos)
```

	Modelo	MSE	R²
0	Random forest 1	6.047060e+08	0.842219
1	Random Forest 2	2.375971e+08	0.910799

In [189…
```
#guardar los modelos
joblib.dump(modelo_rf1, 'RanForest_conOutliner.pkl')
joblib.dump(modelo_rf2, 'RanForest_SinOutliner.pkl')
```

Out[189…
```
['RanForest_SinOutliner.pkl']
```

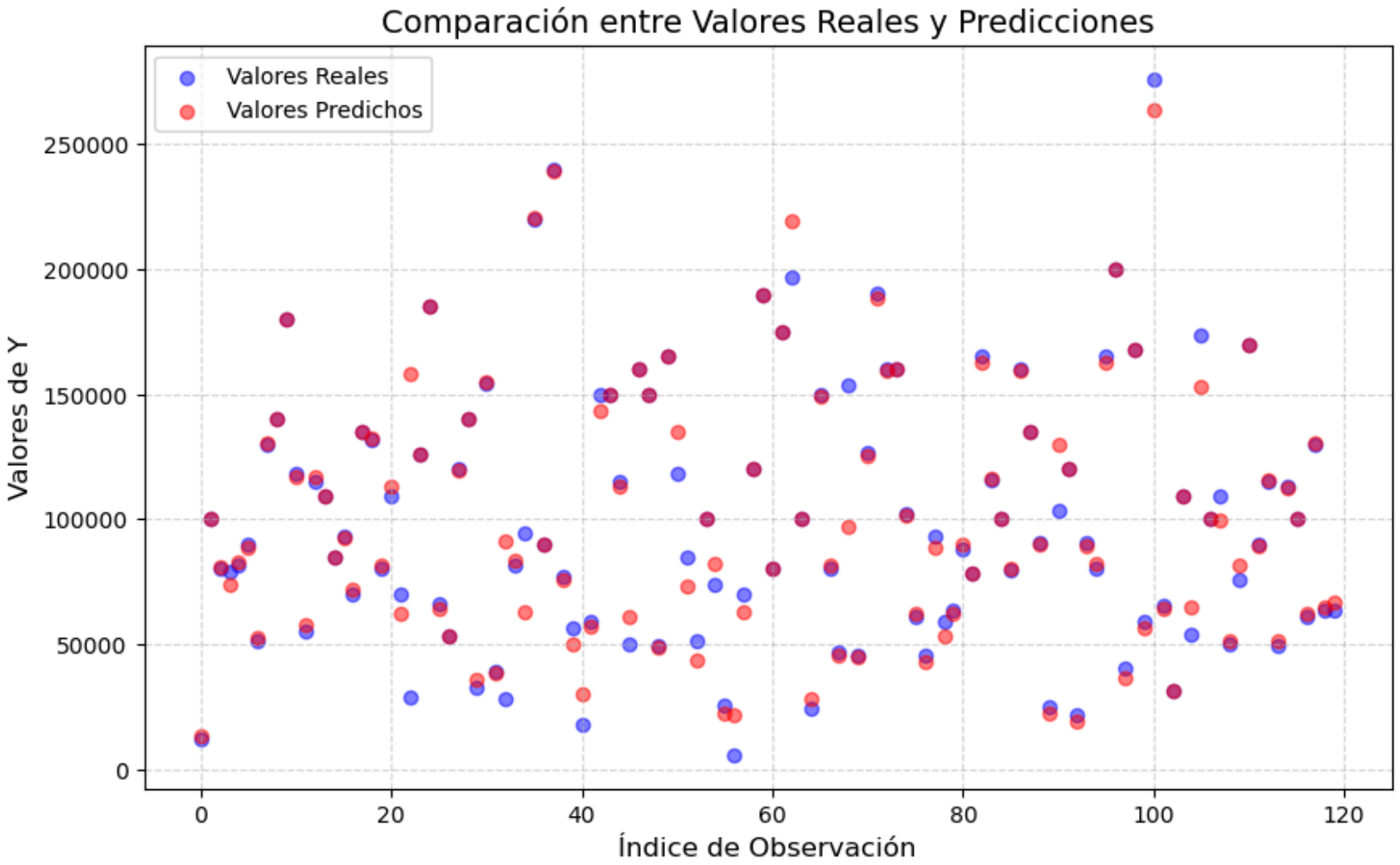
In [191…
```
#mirar puntos
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))

# Graficar cada punto (valor real vs predicho)
plt.scatter(range(len(y_test_4)), y_test_4, color='blue', label="Valores Reales", alpha=0.5)
plt.scatter(range(len(y_pred_rf1)), y_pred_rf1, color='red', label="Valores Predichos", alpha=0.5)



# Configurar el gráfico
plt.xlabel("Índice de Observación", fontsize=12)
plt.ylabel("Valores de Y", fontsize=12)
plt.title("Comparación entre Valores Reales y Predicciones", fontsize=14)
plt.legend()
plt.grid(True, linestyle="--", alpha=0.5)

# Mostrar gráfico
plt.show()
```



In [192…

```
#mirar puntos
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))

# Graficar cada punto (valor real vs predicho)
plt.scatter(range(len(y_test_5)), y_test_5, color='blue', label="Valores Reales", alpha=0.5)
plt.scatter(range(len(y_pred_rf2)), y_pred_rf2, color='red', label="Valores Predichos", alpha=0.5)
```

```
# Configurar el gráfico
plt.xlabel("Índice de Observación", fontsize=12)
plt.ylabel("Valores de Y", fontsize=12)
plt.title("Comparación entre Valores Reales y Predicciones", fontsize=14)
plt.legend()
plt.grid(True, linestyle="--", alpha=0.5)

# Mostrar gráfico
plt.show()
```



interpretación

Error Cuadrático Medio (MSE):

El MSE mide el promedio del cuadrado de los errores (diferencia entre los valores reales y los predichos). Random Forest 2 tiene un MSE mucho menor, lo cual indica que comete menos errores al predecir los valores de salida. Una reducción de ~60% en el error respecto a Random Forest 1 sugiere una mejora significativa en la precisión del modelo. R^2 (Coeficiente de Determinación):

El R^2 indica qué proporción de la variabilidad de los datos es explicada por el modelo. Un R^2 más cercano a 1 indica mejor ajuste. Random Forest 2 tiene un R^2 de 0.91, lo que significa que explica el 91% de la variabilidad de los datos, mientras que Random Forest 1 explica solo el 84%.

In [ ]: