Data salary

Le dataset contient 16534 enregistrements et 11 colonnes :

- **work_year** : Année de travail
- **experience_level** : Niveau d'expérience (SE : Senior, MI : Mid-level, etc.)
- **employment_type** : Type d'emploi (FT : Temps plein, PT : Temps partiel, etc.)
- **job_title** : Titre du poste
- **salary** : Salaire
- **salary_currency** : Devise du salaire
- **salary_in_usd** : Salaire en USD
- **employee_residence** : Résidence de l'employé
- **remote_ratio** : Ratio de travail à distance (0, 50, 100)
- **company_location** : Localisation de l'entreprise
- **company_size** : Taille de l'entreprise (S : Petite, M : Moyenne, L : Grande)

Prétraitement

Installation des bibliothèques

```
[33] !pip install pandas folium
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.1.4)
Requirement already satisfied: folium in /usr/local/lib/python3.10/dist-packages (0.17.0)
Requirement already satisfied: numpy<2,>=1.22.4 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.1)
Requirement already satisfied: branca>=0.6.0 in /usr/local/lib/python3.10/dist-packages (from folium) (0.7.2)
Requirement already satisfied: jinja2>=2.9 in /usr/local/lib/python3.10/dist-packages (from folium) (3.1.4)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from folium) (2.32.3)
Requirement already satisfied: xyzservices in /usr/local/lib/python3.10/dist-packages (from folium) (2024.6.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2>=2.9->folium) (2.1.
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas) (
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->folium)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->folium) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->folium) (2.0.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->folium) (2024
```

Importer les bibliothéques

```
[34] import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import OneHotEncoder, StandardScaler
     from sklearn.linear_model import LinearRegression
     from sklearn.metrics import mean_squared_error
     import folium
     import requests
```

Importer les données

```
[35] df=pd.read_csv("/content/Dataset salary 2024.csv")
     df.head(5)
```

	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_residence
0	2024	SE	FT	AI Engineer	202730	USD	202730	US
1	2024	SE	FT	AI Engineer	92118	USD	92118	US
2	2024	SE	FT	Data Engineer	130500	USD	130500	US
3	2024	SE	FT	Data Engineer	96000	USD	96000	US
4	2024	SE	FT	Machine Learning Engineer	190000	USD	190000	US

```
[36] df.tail(5)
```

	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd
16529	2020	SE	FT	Data Scientist	412000	USD	412000
16530	2021	MI	FT	Principal Data Scientist	151000	USD	151000
16531	2020	EN	FT	Data Scientist	105000	USD	105000
16532	2020	EN	CT	Business Data Analyst	100000	USD	100000
16533	2021	SE	FT	Data Science Manager	7000000	INR	94665

Informations sur le dataframe

```
[38] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16534 entries, 0 to 16533
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   work_year           16534 non-null  int64 
 1   experience_level    16534 non-null  object
 2   employment_type     16534 non-null  object
 3   job_title           16534 non-null  object
 4   salary              16534 non-null  int64 
 5   salary_currency     16534 non-null  object
 6   salary_in_usd       16534 non-null  int64 
 7   employee_residence  16534 non-null  object
 8   remote_ratio        16534 non-null  int64 
 9   company_location    16534 non-null  object
 10  company_size        16534 non-null  object
dtypes: int64(4), object(7)
memory usage: 1.4+ MB
```

Tester les Null

```
[37] df.isnull().sum()
```

	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

dtype: int64

Statistique descriptive

```
[39] summary_stats=df.describe().round(1).transpose()
     summary_stats
```

	count	mean	std	min	25%	50%	75%	max
work_year	16534.0	2023.2	0.7	2020.0	2023.0	2023.0	2024.0	2024.0
salary	16534.0	163727.0	340205.7	14000.0	101763.0	142200.0	187200.0	30400000.0
salary_in_usd	16534.0	149686.8	68505.3	15000.0	101125.0	141300.0	185900.0	800000.0
remote_ratio	16534.0	32.0	46.2	0.0	0.0	0.0	100.0	100.0

```
[42] duplicate_count = df[df.duplicated(subset=['work_year', 'experience_level', 'employment_type', 'job_title', 'salary', 'sa
     print(f"Nombre de lignes dupliquées spécifiques : {duplicate_count}")
```

```
Nombre de lignes dupliquées spécifiques : 6421
```

Supprimer les lignes dupliquées

```
[43] df = df.drop_duplicates(subset=['work_year', 'experience_level', 'employment_type', 'job_title', 'salary', 'salary_curren
```

Re-tester

```
[44] df.duplicated().sum()
```

```
0
```

Tester sur les lignes dupliquées

```
[40] df.duplicated().sum()
```

```
6421
```

Afficher les lignes dupliquées

```
duplicate_rows = df[df.duplicated()]
print(duplicate_rows)
```

```
       work_year experience_level employment_type                 job_title  \
62          2024               SE              FT  Machine Learning Engineer
151         2024               SE              FT           Research Analyst
238         2024               MI              FT               Data Analyst
239         2024               MI              FT               Data Analyst
247         2024               SE              FT             Data Scientist
...          ...              ...             ...                        ...
16235       2022               MI              FT             Data Scientist
16236       2022               SE              FT              Data Engineer
16237       2022               SE              FT              Data Engineer
16376       2021               MI              FT              Data Engineer
16490       2021               MI              FT             Data Scientist
```

Afficher les valeurs unique de chaque colonne

```
[46] def unique_values(data_set):
         for i in data_set:
             print("The unique values for ",i,end="\t")
             print(data_set[i].unique())
     unique_values(df)
```

```
'Applied Research Scientist' 'Prompt Engineer'
'Data Integration Engineer' 'Machine Learning Infrastructure Engineer'
'Data Developer' 'CRM Data Analyst' 'ETL Developer'
'Cloud Database Engineer' 'Data Science Analyst'
'Data Science Practitioner' 'BI Data Analyst' 'Applied Data Scientist'
'Data Quality Engineer' 'Computational Biologist' 'Big Data Engineer'
'Data Analytics Associate' 'Data Reporting Analyst'
'Data Management Consultant' 'Data Quality Analyst'
'Robotics Software Engineer' 'Machine Learning Researcher'
'Data DevOps Engineer' 'Data Science Director' 'Data Strategist'
'Big Data Developer' 'Quantitative Research Analyst'
'Lead Machine Learning Engineer' 'Machine Learning Research Engineer'
'Data Analytics Consultant' 'AI Research Engineer' 'AI Programmer'
'ETL Engineer' 'AI Product Manager' 'AI Developer'
'Computer Vision Engineer' 'Head of Machine Learning' 'Data Analyst Lead'
'Data Integration Developer' 'ML Ops Engineer' 'Data Pipeline Engineer'
'Lead Data Analyst' 'Data Science Lead' 'Director of Data Science'
'Managing Director Data Science' 'Business Data Analyst'
'Marketing Data Scientist' 'Deep Learning Engineer'
'Machine Learning Modeler' 'Decision Scientist' 'Financial Data Analyst'
'Data Strategy Manager' 'Data Visualization Engineer'
'Azure Data Engineer' 'Principal Data Scientist' 'Staff Data Analyst'
```

Compter le nombre d'employeurs chaque année

```
[47] work_year_counts = df.work_year.value_counts().sort_index()
     print(work_year_counts)
```

```
work_year
2020      75
2021     216
2022    1116
2023    4632
2024    4074
Name: count, dtype: int64
```

Dictionnaire des codes de pays et leurs noms complets

```
#Dictionnaire des codes de pays et leurs noms complets
country_names = {
    'US': 'United States', 'AU': 'Australia', 'GB': 'United Kingdom', 'CA': 'Canada',
    'NL': 'Netherlands', 'LT': 'Lithuania', 'DK': 'Denmark', 'FR': 'France',
    'ZA': 'South Africa', 'NZ': 'New Zealand', 'AR': 'Argentina', 'ES': 'Spain',
    'KE': 'Kenya', 'LV': 'Latvia', 'GE': 'Georgia', 'IN': 'India', 'DE': 'Germany',
    'IL': 'Israel', 'FI': 'Finland', 'AT': 'Austria', 'HR': 'Croatia', 'BR': 'Brazil',
    'CH': 'Switzerland', 'AE': 'United Arab Emirates', 'GR': 'Greece', 'PL': 'Poland',
    'SA': 'Saudi Arabia', 'UA': 'Ukraine', 'EG': 'Egypt', 'PH': 'Philippines',
    'TR': 'Turkey', 'OM': 'Oman', 'MX': 'Mexico', 'PT': 'Portugal', 'BA': 'Bosnia and Herzegovina',
    'IT': 'Italy', 'IE': 'Ireland', 'EE': 'Estonia', 'MT': 'Malta', 'LB': 'Lebanon',
    'RO': 'Romania', 'HU': 'Hungary', 'VN': 'Vietnam', 'NG': 'Nigeria', 'CZ': 'Czech Republic',
    'PK': 'Pakistan', 'UG': 'Uganda', 'CO': 'Colombia', 'SI': 'Slovenia', 'MU': 'Mauritius',
    'AM': 'Armenia', 'TH': 'Thailand', 'KR': 'South Korea', 'QA': 'Qatar', 'RU': 'Russia',
    'TN': 'Tunisia', 'GH': 'Ghana', 'BE': 'Belgium', 'AD': 'Andorra', 'EC': 'Ecuador',
    'PE': 'Peru', 'MD': 'Moldova', 'NO': 'Norway', 'UZ': 'Uzbekistan', 'JP': 'Japan',
    'HK': 'Hong Kong', 'CF': 'Central African Republic', 'SG': 'Singapore', 'SE': 'Sweden',
    'KW': 'Kuwait', 'CY': 'Cyprus', 'IR': 'Iran', 'AS': 'American Samoa', 'CN': 'China',
    'CR': 'Costa Rica', 'CL': 'Chile', 'PR': 'Puerto Rico', 'BO': 'Bolivia', 'DO': 'Dominican Republic',
    'ID': 'Indonesia', 'MY': 'Malaysia', 'HN': 'Honduras', 'DZ': 'Algeria', 'IQ': 'Iraq',
    'BG': 'Bulgaria', 'JE': 'Jersey', 'RS': 'Serbia', 'LU': 'Luxembourg'
}
```

```
[48] # Remplacement des codes par les noms complets
     df['employee_residence'] = df['employee_residence'].map(country_names)
     df.head(10)
```

```
<ipython-input-48-68a493b2e653>:25: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
  df['employee_residence'] = df['employee_residence'].map(country_names)
```

	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_residence
0	2024	SE	FT	AI Engineer	202730	USD	202730	United States
1	2024	SE	FT	AI Engineer	92118	USD	92118	United States
2	2024	SE	FT	Data Engineer	130500	USD	130500	United States
3	2024	SE	FT	Data Engineer	96000	USD	96000	United States
4	2024	SE	FT	Machine Learning Engineer	190000	USD	190000	United States
5	2024	SE	FT	Machine Learning Engineer	160000	USD	160000	United States

Carte des emplois

```
# Compter le nombre d'emplois par pays
job_count_by_country = df['employee_residence'].value_counts().reset_index()
job_count_by_country.columns = ['country', 'job_count']
# Déterminer le poste le plus commun (top emploi) pour chaque pays
top_job_by_country = df.groupby('employee_residence')['job_title'].agg(lambda x: x.value_counts().idxmax()).reset_index()
top_job_by_country.columns = ['country', 'top_job']

# Fusionner les DataFrames pour avoir à la fois le nombre d'emplois et le top emploi par pays
job_summary_by_country = pd.merge(job_count_by_country, top_job_by_country, on='country')

# Télécharger le fichier GeoJSON des frontières des pays
url = 'https://raw.githubusercontent.com/johan/world.geo.json/master/countries.geo.json'
geo_json_data = requests.get(url).json()

# Créer la carte
world_map = folium.Map(location=[20, 0], zoom_start=2)

# Ajouter le choroplèthe
folium.Choropleth(
    geo_data=geo_json_data,
    name='choropleth',
    data=job_count_by_country,
    columns=['country', 'job_count'],
    key_on='feature.properties.name',
```

[53]

```
    fill_color='YlGnBu',
    fill_opacity=0.7,
    line_opacity=0.2,
    legend_name='Number of Data Jobs'
).add_to(world_map)

# Ajouter les marqueurs pour les emplois individuels
country_coords = {
    'United States': [37.0902, -95.7129], 'Australia': [-25.2744, 133.7751], 'United Kingdom': [55.3781, -3.4360],
    'Canada': [56.1304, -106.3468], 'Netherlands': [52.1326, 5.2913], 'Lithuania': [55.1694, 23.8813],
    'Denmark': [56.2639, 9.5018], 'France': [46.6034, 1.8883], 'South Africa': [-30.5595, 22.9375],
    'New Zealand': [-40.9006, 174.8860], 'Argentina': [-38.4161, -63.6167], 'Spain': [40.4637, -3.7492],
    'Kenya': [-0.0236, 37.9062], 'Latvia': [56.8796, 24.6032], 'Georgia': [42.3154, 43.3569],
    'India': [20.5937, 78.9629], 'Germany': [51.1657, 10.4515], 'Israel': [31.0461, 34.8516],
    'Finland': [61.9241, 25.7482], 'Austria': [47.5162, 14.5501], 'Croatia': [45.1, 15.2],
    'Brazil': [-14.2350, -51.9253], 'Switzerland': [46.8182, 8.2275], 'United Arab Emirates': [23.4241, 53.8478],
    'Greece': [39.0742, 21.8243], 'Poland': [51.9194, 19.1451], 'Saudi Arabia': [23.8859, 45.0792],
    'Ukraine': [48.3794, 31.1656], 'Egypt': [26.8206, 30.8025], 'Philippines': [12.8797, 121.7740],
    'Turkey': [38.9637, 35.2433], 'Oman': [21.4735, 55.9754], 'Mexico': [23.6345, -102.5528],
    'Portugal': [39.3999, -8.2245], 'Bosnia and Herzegovina': [43.9159, 17.6791], 'Italy': [41.8719, 12.5674],
    'Ireland': [53.1424, -7.6921], 'Estonia': [58.5953, 25.0136], 'Malta': [35.9375, 14.3754],
    'Lebanon': [33.8547, 35.8623], 'Romania': [45.9432, 24.9668], 'Hungary': [47.1625, 19.5033],
    'Vietnam': [14.0583, 108.2772], 'Nigeria': [9.0820, 8.6753], 'Czech Republic': [49.8175, 15.4730],
    'Pakistan': [30.3753, 69.3451], 'Uganda': [1.3733, 32.2903], 'Colombia': [4.5709, -74.2973],
    'Slovenia': [46.1512, 14.9955], 'Mauritius': [-20.3484, 57.5522], 'Armenia': [40.0691, 45.0382],
    'Thailand': [15.8700, 100.9925], 'South Korea': [35.9078, 127.7669], 'Qatar': [25.3548, 51.1839],
    'Russia': [61.5240, 105.3188], 'Tunisia': [33.8869, 9.5375], 'Ghana': [7.9465, -1.0232],
    'Belgium': [50.8503, 4.3517], 'Andorra': [42.5063, 1.5218], 'Ecuador': [-1.8312, -78.1834],
    'Peru': [-9.1900, -75.0152], 'Moldova': [47.4116, 28.3699], 'Norway': [60.4720, 8.4689],
    'Uzbekistan': [41.3775, 64.5853], 'Japan': [36.2048, 138.2529], 'Hong Kong': [22.3193, 114.1694],
```

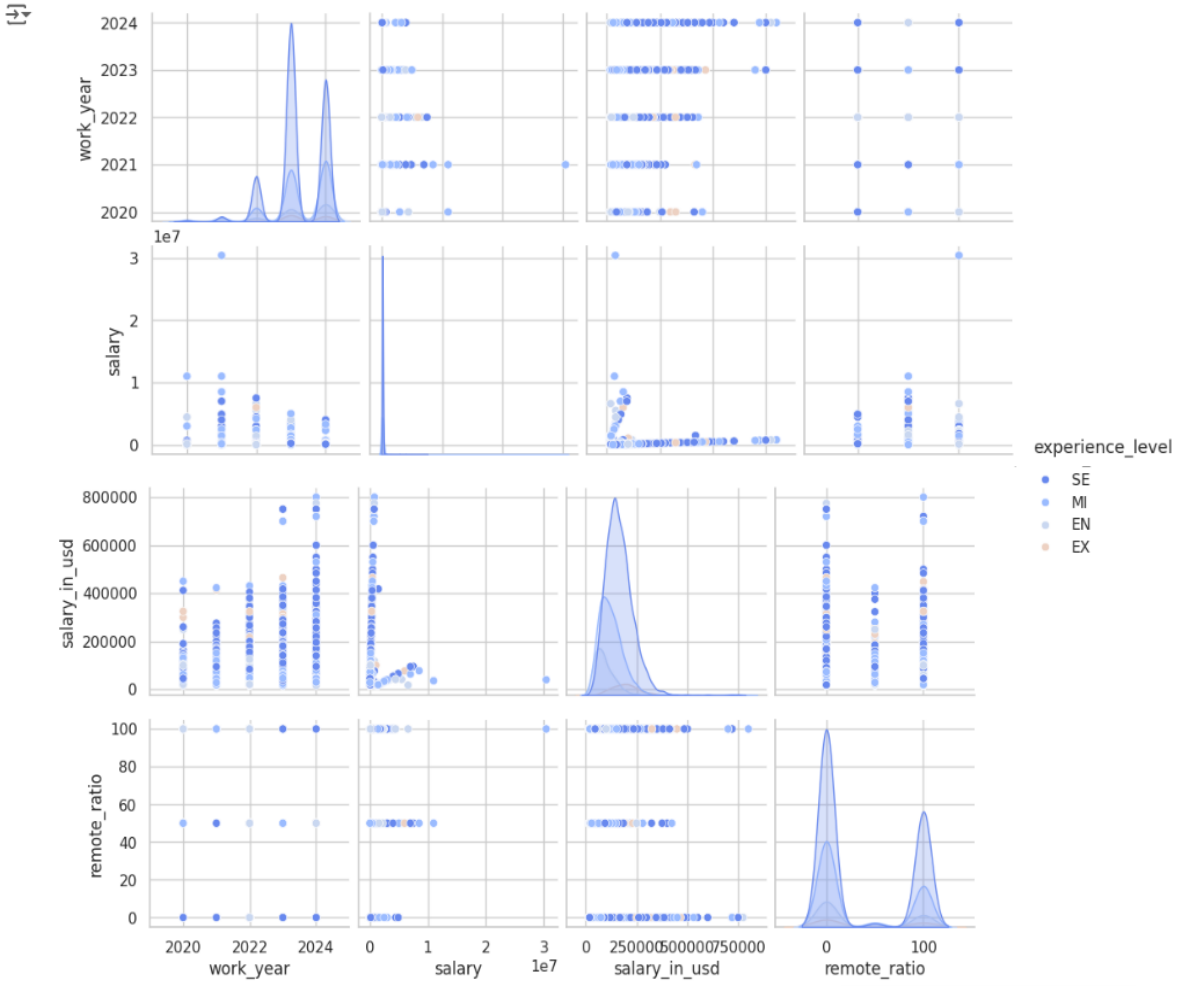
```
[53]      'Central African Republic': [6.6111, 20.9394], 'Singapore': [1.3521, 103.8198], 'Sweden': [60.1282, 18.6435],
          'Kuwait': [29.3759, 47.9774], 'Cyprus': [35.1264, 33.4299], 'Iran': [32.4279, 53.6880],
          'American Samoa': [-14.2710, -170.1322], 'China': [35.8617, 104.1954], 'Costa Rica': [9.7489, -83.7534],
          'Chile': [-35.6751, -71.5430], 'Puerto Rico': [18.2208, -66.5901], 'Bolivia': [-16.2902, -63.5887],
          'Dominican Republic': [18.7357, -70.1627], 'Indonesia': [-0.7893, 113.9213], 'Malaysia': [4.2105, 101.9758],
          'Honduras': [15.2000, -86.2419], 'Algeria': [28.0339, 1.6596], 'Iraq': [33.2232, 43.6793],
          'Bulgaria': [42.7339, 25.4858], 'Jersey': [49.2144, -2.1313], 'Serbia': [44.0165, 21.0059],
          'Luxembourg': [49.8153, 6.1296]
      }

      for index, row in df.iterrows():
          country = row['employee_residence']
          if country in country_coords:
              # Récupérer le nombre d'emplois et le top emploi pour le pays actuel
              job_count = job_summary_by_country[job_summary_by_country['country'] == country]['job_count'].values[0]
              top_job = job_summary_by_country[job_summary_by_country['country'] == country]['top_job'].values[0]

              folium.Marker(
                  location=country_coords[country],
                  popup=f"Top emploi: {top_job}\nNb job data: {job_count}",
                  icon=folium.Icon(icon='briefcase', color='blue')
              ).add_to(world_map)
```

Affichage de Carte

```
world_map
```

Visualisation des relations entre les variables en fonction du niveau d'expérience avec Pairplot

```
[56] sns.pairplot(df, hue='experience_level')
     plt.show()
```

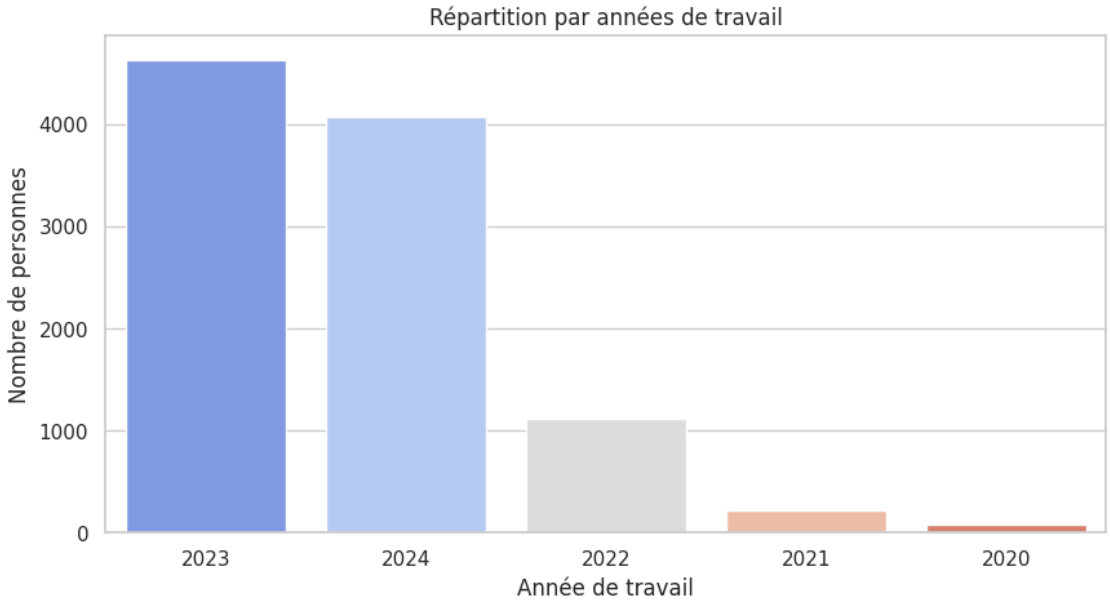


Boxplot des salaires en fonction du niveau d'expérience

```
[59] plt.figure(figsize=(7, 3))
     sns.boxplot(x='experience_level', y='salary_in_usd', data=df)
     plt.title('Salary by Experience Level')
     plt.xlabel('Experience Level')
     plt.ylabel('Salary in USD')
     plt.show()
```

Répartition des employés par année

```
[61] # Données de l'exemple
     work_year_counts = df['work_year'].value_counts()

     # Définir la palette de couleurs 'coolwarm' de Seaborn
     sns.set(style="whitegrid", palette="coolwarm")

     # Créer le graphique avec Seaborn
     plt.figure(figsize=(10, 5))
     sns.countplot(x='work_year', data=df, palette='coolwarm', order=work_year_counts.index)
     plt.title('Répartition par années de travail')
     plt.xlabel('Année de travail')
     plt.ylabel('Nombre de personnes')
     plt.xticks(rotation=0)
     plt.show()
```



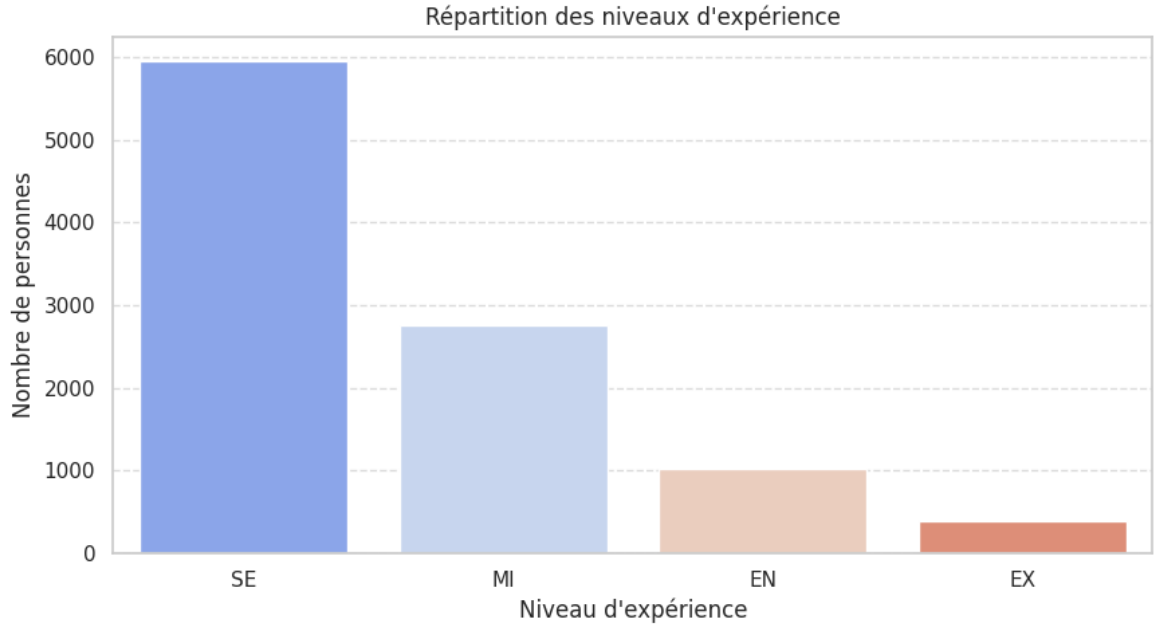
Comptage des occurrences de chaque intitulé de poste

```
[26] job_title_counts = df.job_title.value_counts()
     print(job_title_counts)
```

```
job_title
Data Engineer                    1842
Data Scientist                   1835
Data Analyst                     1384
Machine Learning Engineer         945
Analytics Engineer                354
                                 ...
Quantitative Research Analyst       1
AWS Data Architect                  1
Analytics Engineering Manager       1
Marketing Data Scientist            1
Applied Research Scientist          1
Name: count, Length: 155, dtype: int64
```

Répartition des niveaux d'expérience avec un diagramme à barres

```
[27] # Créer un diagramme à barres pour visualiser la répartition des niveaux d'expérience avec une palette de couleurs de cha
     plt.figure(figsize=(10, 6))
     sns.countplot(x='experience_level', data=df, palette='coolwarm')
     plt.xlabel('Niveau d\'expérience')
     plt.ylabel('Nombre de personnes')
     plt.title('Répartition des niveaux d\'expérience')
     plt.grid(axis='y', linestyle='--', alpha=0.7)

     plt.show()
```

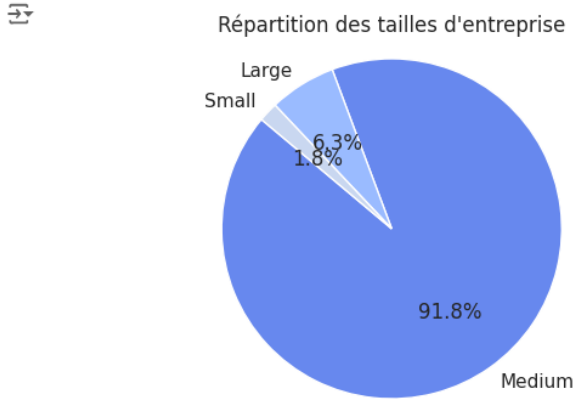


Répartition des tailles d'entreprise

```
[28] sizes=df.company_size.value_counts()
     print(sizes)
```

```
company_size
M    9284
L     642
S     187
Name: count, dtype: int64
```

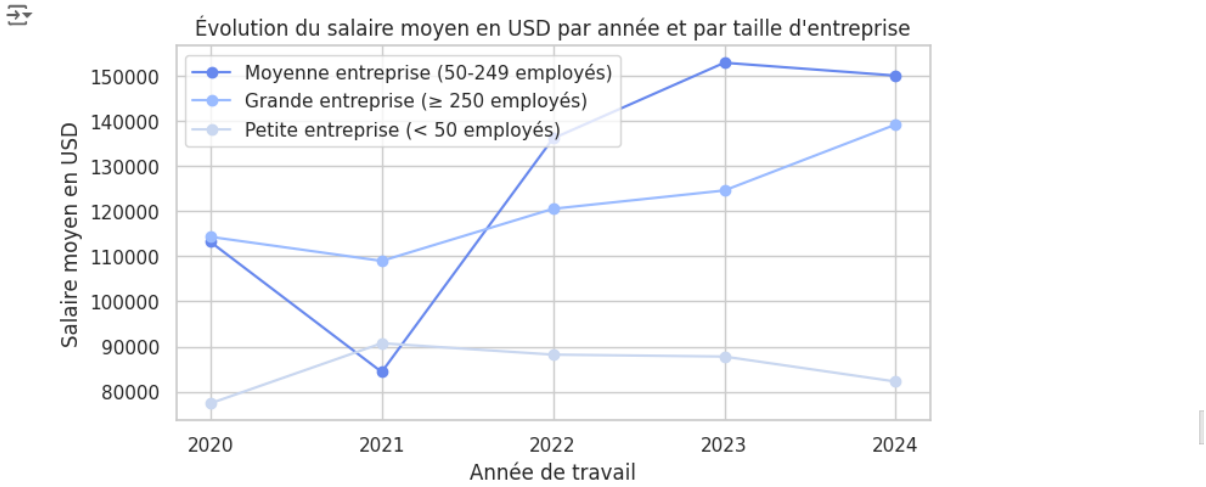
```
[64] labels = ['Medium', 'Large', 'Small']  # Étiquettes correspondant à 'M', 'L', 'S'
     colors = sns.color_palette('coolwarm')  # Palette de couleurs 'coolwarm' de Seaborn

     # Créer le diagramme circulaire avec Matplotlib
     plt.figure(figsize=(7, 4))
     plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140)
     plt.title('Répartition des tailles d\'entreprise')
     plt.axis('equal')  # Assure un cercle parfait

     plt.show()
```



Évolution du salaire moyen en USD par année et par taille d'entreprise

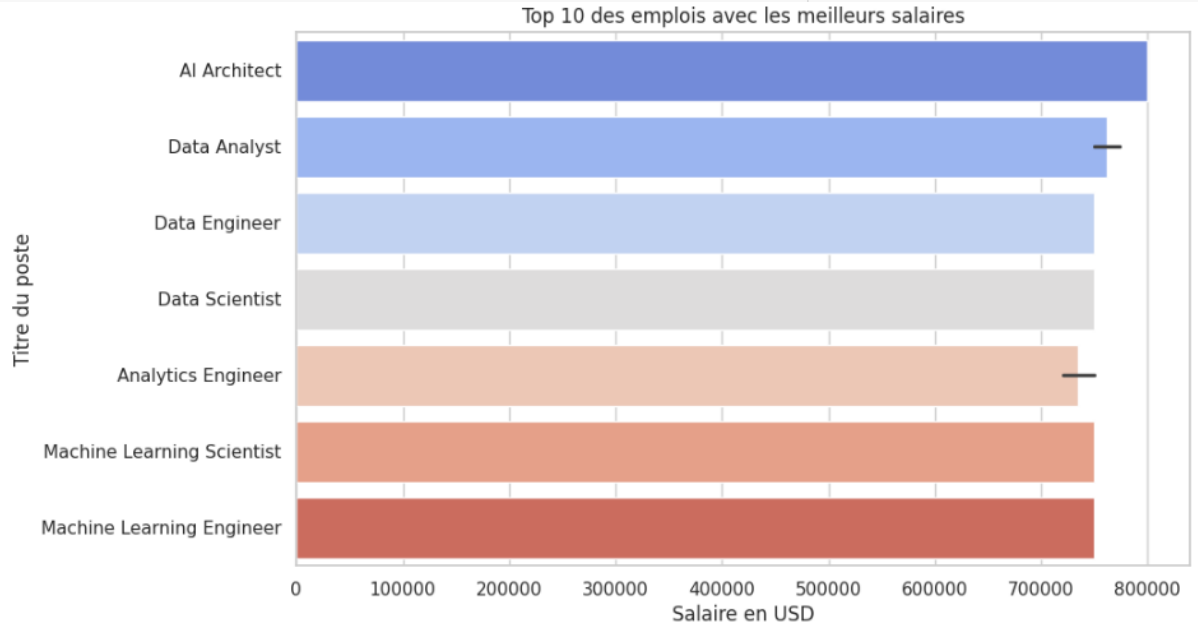
```
[30] # Convertir work_year en entier pour enlever les décimales
     df['work_year'] = df['work_year'].astype(int)

     # Calculer la moyenne des salaires en USD par année et par company_size
     average_salary_usd_by_year_company_size = df.groupby(['work_year', 'company_size'])['salary_in_usd'].mean().reset_index()

     # Définir les tailles d'entreprise (company_size) à afficher
     company_sizes = df['company_size'].unique()
     size_mapping = {
         'S': 'Petite entreprise (< 50 employés)',
         'M': 'Moyenne entreprise (50-249 employés)',
         'L': 'Grande entreprise (≥ 250 employés)'
     }
     # Créer le graphique avec Matplotlib en utilisant plusieurs lignes
     plt.figure(figsize=(10, 6))

     for size in company_sizes:
         subset = average_salary_usd_by_year_company_size[average_salary_usd_by_year_company_size['company_size'] == size]
         plt.plot(subset['work_year'], subset['salary_in_usd'], marker='o', linestyle='-', label=size_mapping[size])

     plt.title('Évolution du salaire moyen en USD par année et par taille d\'entreprise')
```

```
[65] plt.legend()
     plt.grid(True)

     # Obtenir les années uniques après conversion en entier
     years = average_salary_usd_by_year_company_size['work_year'].unique()
     plt.xticks(years)  # Définir les années sur l'axe des x

     plt.show()
```



Top 10 des emplois avec les meilleurs salaires en USD

```
[31] # Trier le DataFrame par salaire en USD de manière décroissante
     df_sorted = df.sort_values(by='salary_in_usd', ascending=False)

     # Sélectionner les 10 premiers enregistrements
     top_10_jobs = df_sorted.head(10)

     # Visualisation des 10 meilleurs emplois avec un graphe
     plt.figure(figsize=(10, 6))
     sns.barplot(x='salary_in_usd', y='job_title', data=top_10_jobs, palette="coolwarm")
     plt.xlabel('Salaire en USD')
     plt.ylabel('Titre du poste')
     plt.title('Top 10 des emplois avec les meilleurs salaires')
     plt.show()
```



Répartition des 10 emplois les plus fréquents

```
[32] # Assuming 'df' is your DataFrame and 'job_title' is a column in it
     data = df['job_title'].value_counts().head(10)

     # Create a new figure for the specific plot
     fig, ax = plt.subplots()
     ax.pie(data, labels=data.index, autopct='%1.1f%%')
     ax.set_title('Job Title')

     # Display the plot
     plt.show()
```