Employee Salaries for different job roles

Alejandro Borrego Megías

19/10/2023

Introducción

En este proyecto se desarrolla en Python un análisis básico de datos sobre los sueldos que ganan distintos empleados según sus cometidos y experiencia, a lo largo de distintos negocios y zonas del mundo. La URL de referencia es la siguiente:

https://www.kaggle.com/datasets/inductiveanks/employee-salaries-for-different-job-roles

En ella puede encontrarse información más detallada, así como una descripción precisa de cada columna. Seguidamente, te toca a ti hacer una breve introducción, completando el fragmento de letra en azul y desarrollándolo a tu antojo.

A partir de los datos proporcionados, he conseguido afianzar mis conocimientos sobre el lenguaje Python en el manejo de estructuras de datos como diccionarios, listas, conjuntos o dataframes de pandas. Por otro lado he ampliado mis conocimientos empleando por primera técnicas de map-reduce para un caso práctico con la librería `mrjob` y en la representación de gráficos con la librería `matplotlib`. Por otro lado, por falta de tiempo debido a la carga de trabajo del master, no he conseguido completar el ejercicio *G* como me hubiera gustado, me habría gustado probar otros modelos y técnicas y compararlos.

Aunque al final de este notebook detallaré la calificación que calculo honestamente, globalmente, siguiendo las puntuaciones que se asigna a cada apartado, diría que he obtenido una nota de 10 sobre 10.

Librerías

Pongamos todas las librerías necesarias al principio, tal como propone el estilo `pep-8`. Ej.:

PEP 8 -- Style Guide for Python Code.

In [1]:

```
# Imports del módulo estándar de Python
from collections import defaultdict
from typing import List, Dict, Tuple, Optional

# Imports de terceros
import csv
import numpy as np
import pandas as pd
from matplotlib.ticker import MaxNLocator
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import SGDRegressor

import ipytest
```

a) Algunas operaciones sencillas [3 puntos]

Nuestra tabla de datos es un archivo de texto (`ds_salaries.csv`) que puede verse así con cualquier editor:

No description has been provided for this image

La primera columna es la cabecera, y contiene los nombres de los campos, separados por comas. Las demás, son los valores de dichos campos, consignando los datos de cada vehículo en una línea.

Si la abrimos con *excell*, vemos cada línea en una celda, sin separar los distintos campos:

No description has been provided for this image

a.1) Cambiar el formato del archivo `csv` a "punto y coma"

Podemos importar la tabla de datos desde excell (pestaña `datos`), simplemente indicando que el separador es una coma:

No description has been provided for this image

Pero te propongo generar un archivo como el anterior, pero que use el punto y coma como separador, en vez de la coma:

No description has been provided for this image

Para ello, debes diseñar una función que tome con un archivo como el de partida que usa la coma como separador, y genere otro, con el punto y coma como separador.

In [2]:
```python
# Esta celda debe ser completada por el estudiante.
def to_semicolon (input_file_path : str, output_file_path: str):
    """
    Replace all commas with semicolons in the
    content of a CSV file and write the modified content to another file.

    Parameters
    ----------
    input_file_path, output_file_path: str
    """
    try:
        with open(input_file_path, 'r', newline='') as csvfile_input:
            csv_content = csvfile_input.read()
            csv_to_pc_content = csv_content.replace(',', ';')
    except FileNotFoundError:
        print(f"File not found: {input_file_path}")
    except Exception as e:
        print(f"An error occurred: {e}")

    try:
        with open(output_file_path, 'w') as csvfile_output:
            csvfile_output.write(csv_to_pc_content)
    except FileNotFoundError:
        print(f"File not found: {input_file_path}")
    except Exception as e:
        print(f"An error occurred: {e}")
```

In [3]:
```python
# Ejecución de la función anterior:

DatosComas = "ds_salaries.csv"
DatosPunComas = "ds_salaries_pc.csv"
to_semicolon(DatosComas, DatosPunComas)
```

In [4]:
```python
# Comprobamos que funciona como es debido, viendo las primeras cinco filas de ambos

with open(DatosComas, "r") as f:
    for _ in range(5):
        linea = f.readline()
        print(linea)

print("....................................")

with open(DatosPunComas, "r") as f:
    for _ in range(5):
        linea = f.readline()
        print(linea)
```

```
,work_year,experience_level,employment_type,job_title,salary,salary_currency,salary_
in_usd,employee_residence,remote_ratio,company_location,company_size

0,2020,MI,FT,Data Scientist,70000,EUR,79833,DE,0,DE,L

1,2020,SE,FT,Machine Learning Scientist,260000,USD,260000,JP,0,JP,S

2,2020,SE,FT,Big Data Engineer,85000,GBP,109024,GB,50,GB,M

3,2020,MI,FT,Product Data Analyst,20000,USD,20000,HN,0,HN,S

....................................
;work_year;experience_level;employment_type;job_title;salary;salary_currency;salary_
in_usd;employee_residence;remote_ratio;company_location;company_size

0;2020;MI;FT;Data Scientist;70000;EUR;79833;DE;0;DE;L

1;2020;SE;FT;Machine Learning Scientist;260000;USD;260000;JP;0;JP;S

2;2020;SE;FT;Big Data Engineer;85000;GBP;109024;GB;50;GB;M

3;2020;MI;FT;Product Data Analyst;20000;USD;20000;HN;0;HN;S
```

Nota. En la comprobación anterior, por cada línea que se imprime con la instrucción `print`, se realizan dos saltos de línea. Eso es porque las líneas anteriores se han cargado con la marca `\n`, como puedes ver a continuación, con la última línea. En las funciones que siguen deberás tener esto en cuenta para suprimir la marca `\n` cuando sea necesario.

In [5]:
```
#Observa la marca "\n" al final de la última línea leída:

linea
```

Out[5]:
```
'3;2020;MI;FT;Product Data Analyst;20000;USD;20000;HN;0;HN;S\n'
```

a.2) Selección de una línea, separando sus campos

Diseña ahora una función que selecciona una línea y nos da una lista con los valores de sus campos. Los ejemplares de funcionamiento te darán la información sobre cómo deseamos que funcione:

In [6]:
```
# Esta celda debe ser completada por el estudiante
def select_line(input_file_path: str, line_number: int) -> List[str]:
    """
    Given a file path returns a specific line

    Parameters
    ----------
    input_file_path: str
    line_number: int

    Returns
```

```
    -------
    list(str)
        Returns the line in file input_file_path in the position line_number
    """
    try:
        with open(input_file_path, 'r', newline='') as csvfile_input:
            csv_content = csvfile_input.readlines()
            return csv_content[line_number].strip().split(";")
    except FileNotFoundError:
        print(f"File not found: {input_file_path}")
        return []
    except Exception as e:
        print(f"An error occurred: {e}")
        return []
```

In [7]:
```
# Comprobación del funcionamiento:

cabecera = select_line(DatosPunComas, 0)
print(cabecera)

linea_1 = select_line(DatosPunComas, 1)
print(linea_1)
```

```
['', 'work_year', 'experience_level', 'employment_type', 'job_title', 'salary', 'sal
ary_currency', 'salary_in_usd', 'employee_residence', 'remote_ratio', 'company_locat
ion', 'company_size']
['0', '2020', 'MI', 'FT', 'Data Scientist', '70000', 'EUR', '79833', 'DE', '0', 'D
E', 'L']
```

Nota: Observa que se suprime la marca de fin de línea, `\n`.

a.3) Ajustes en nuestro archivo de datos

En el archivo de datos, podemos prescindir de la primera fila, que es la cabecera, y de la primera columna, pues únicamente da un número de orden de las filas, de manera que vamos a suprimir ambas, la primera fila y la primera columna; también, la columna de la experiencia será más manejable si convertimos los código en números (así: "EN" -> 0, "MI" -> 1, "EX" -> 2, "SE" -> 3) y algo parecido haremos con el tamaño de las compañías ("S" -> 1, "EX: 0, "M" -> 2, "L" -> 3). Finalmente, para nuestros fines, preferimos manejar el salario en una moneda común, de manera que descartamos las columnas relativas al sueldo en las monedas de cada país y retenemos únicamente la que refleja el salario en dólares.

Realiza estos cambios y, con ellos, genera el archivo nuevo: `DatosSalariosNormalizados.csv`.

In [8]:
```
# Esta celda debe ser completada por el estudiante

def normalize_data(input_file_path: str, output_file_path: str):
    """
    Reads a CSV file, normalizes its data, and writes the result to a new CSV file.
```

```
    This function performs the following data normalization steps:
    1. Reads the CSV file specified by input_file_path.
    2. Maps the 'experience_level' and 'company_size' values using predefined dicti
    3. Removes the 'salary' and 'salary_currency' columns from the data.
    4. Writes the resulting data to a new CSV file specified by output_file_path.

    Parameters
    ----------
    input_file_path : str
        The path to the input CSV file.

    output_file_path : str
        The path to the output CSV file where the normalized data will be saved.
    """
    try:
        # Define mapping dictionaries
        experience_mapping = {"EN": 0, "MI": 1, "EX": 2, "SE": 3}
        company_size_mapping = {"S": 1, "EX": 0, "M": 2, "L": 3}

        with open(input_file_path, 'r', newline='') as infile, open(output_file_pat
            reader = infile.readlines()
            writer = csv.writer(outfile, delimiter=';')
            first_line = True
            experience_level_index = 0
            company_size_index = 0
            salary_index = 0
            salary_currency_index = 0
            for line in reader:
                row = line.strip().split(',')
                if first_line:
                    # Get special indices
                    experience_level_index = row.index('experience_level')
                    company_size_index = row.index('company_size')
                    salary_index = row.index('salary')
                    salary_currency_index = row.index('salary_currency')
                    first_line = False
                else:
                    # Extract the experience level and company size columns
                    experience_level = row[experience_level_index]
                    company_size = row[company_size_index]

                    # Normalize experience level and company size
                    experience_level = str(experience_mapping.get(experience_level,
                    company_size = str(company_size_mapping.get(company_size, compa

                    # Remove salary and salary_currency columns
                    normalized_row = []
                    for index in range(len(row)):
                        if index == experience_level_index:
                            normalized_row.append(experience_level)
                        elif index == company_size_index:
                            normalized_row.append(company_size)
                        elif index == 0 or index == salary_index or index == salary
                            # we skip this values
                            continue
                        else:
```

```python
                            normalized_row.append(row[index])

                    # Write the normalized row to the output file
                    writer.writerow(normalized_row)
    except FileNotFoundError:
        print(f"File '{input_file_path}' not found.")
    except Exception as e:
        print(f"An error occurred: {str(e)}")
```

In [9]:
```python
# Comprobación de funcionamiento:

DatosSalariosNormalizados = "ds_salaries.norm.csv"
normalize_data(DatosComas, DatosSalariosNormalizados)
```

En este apartado se ha conseguido completar la totalidad de los ejercicios obteniendo los resultados esperados por el profesor siguiendo la metodología de desarrollo TDD, usando como tests la salida proporcionada en el notebook.

b) extracción de algunos datos globales directamente de los archivos [2 puntos]

b.1) Relación de puestos y su frecuencia

Con el archivo de datos normalizado, deseamos conocer la relación de los cargos que aparecen en el archivo, así como su frecuencia.

In [10]:
```python
def puesto_frec(input_file_path: str) -> dict:
    """
    Reads a CSV file and calculates the frequencies of job positions.

    This function reads the specified CSV file and counts the frequencies of job po

    Parameters
    ----------
    input_file_path : str
        The path to the input CSV file.

    Returns
    -------
    dict
        A dictionary containing the job positions (puestos) as keys and their frequ
    """
    # Initialize a dictionary to store job frequencies
    frequencies = {}

    try:
        with open(input_file_path, 'r', newline='') as infile:
            reader = infile.readlines()
            for line in reader:
                row = line.strip().split(';')
```

```
                job_position = row[3]  # Assuming the job position is in the 4th co

                # Count the frequency of each job position
                if job_position in frequencies:
                    frequencies[job_position] += 1
                else:
                    frequencies[job_position] = 1

        return frequencies
    except FileNotFoundError:
        print(f"File '{input_file_path}' not found.")
    except Exception as e:
        print(f"An error occurred: {str(e)}")
```

In [11]:
```
# Comprobación de funcionamiento:

puesto_y_frec = puesto_frec(DatosSalariosNormalizados)
puesto_y_frec
```

Out[11]:
```
{'Data Scientist': 143,
 'Machine Learning Scientist': 8,
 'Big Data Engineer': 8,
 'Product Data Analyst': 2,
 'Machine Learning Engineer': 41,
 'Data Analyst': 97,
 'Lead Data Scientist': 3,
 'Business Data Analyst': 5,
 'Lead Data Engineer': 6,
 'Lead Data Analyst': 3,
 'Data Engineer': 132,
 'Data Science Consultant': 7,
 'BI Data Analyst': 6,
 'Director of Data Science': 7,
 'Research Scientist': 16,
 'Machine Learning Manager': 1,
 'Data Engineering Manager': 5,
 'Machine Learning Infrastructure Engineer': 3,
 'ML Engineer': 6,
 'AI Scientist': 7,
 'Computer Vision Engineer': 6,
 'Principal Data Scientist': 7,
 'Data Science Manager': 12,
 'Head of Data': 5,
 '3D Computer Vision Researcher': 1,
 'Data Analytics Engineer': 4,
 'Applied Data Scientist': 5,
 'Marketing Data Analyst': 1,
 'Cloud Data Engineer': 2,
 'Financial Data Analyst': 2,
 'Computer Vision Software Engineer': 3,
 'Director of Data Engineering': 2,
 'Data Science Engineer': 3,
 'Principal Data Engineer': 3,
 'Machine Learning Developer': 3,
 'Applied Machine Learning Scientist': 4,
 'Data Analytics Manager': 7,
 'Head of Data Science': 4,
 'Data Specialist': 1,
 'Data Architect': 11,
 'Finance Data Analyst': 1,
 'Principal Data Analyst': 2,
 'Big Data Architect': 1,
 'Staff Data Scientist': 1,
 'Analytics Engineer': 4,
 'ETL Developer': 2,
 'Head of Machine Learning': 1,
 'NLP Engineer': 1,
 'Lead Machine Learning Engineer': 1,
 'Data Analytics Lead': 1}
```

b.2) Ídem, usando diccionarios por defecto

In [12]:
```
def puesto_frec(input_file_path: str) -> defaultdict[int]:
    """
    Reads a CSV file and calculates the frequencies of job positions.

    This function reads the specified CSV file and counts the frequencies of job po

    Parameters
    ----------
    input_file_path : str
        The path to the input CSV file.

    Returns
    -------
    defaultdict
        A defaultdict with job positions (puestos) as keys and their frequencies as
    """
    # Initialize a defaultdict to store job frequencies as integers
    frequencies = defaultdict(int)

    try:
        with open(input_file_path, 'r', newline='') as infile:
            reader = infile.readlines()
            for line in reader:
                row = line.strip().split(';')
                job_position = row[3]  # Assuming the job position is in the 4th co

                # Count the frequency of each job position
                frequencies[job_position] += 1

        return frequencies
    except FileNotFoundError:
        print(f"File '{input_file_path}' not found.")
    except Exception as e:
        print(f"An error occurred: {str(e)}")
```

In [13]:
```
# Comprobación de funcionamiento:

puesto_y_frec = puesto_frec(DatosSalariosNormalizados)
puesto_y_frec
```

```
Out[13]: defaultdict(int,
                {'Data Scientist': 143,
                 'Machine Learning Scientist': 8,
                 'Big Data Engineer': 8,
                 'Product Data Analyst': 2,
                 'Machine Learning Engineer': 41,
                 'Data Analyst': 97,
                 'Lead Data Scientist': 3,
                 'Business Data Analyst': 5,
                 'Lead Data Engineer': 6,
                 'Lead Data Analyst': 3,
                 'Data Engineer': 132,
                 'Data Science Consultant': 7,
                 'BI Data Analyst': 6,
                 'Director of Data Science': 7,
                 'Research Scientist': 16,
                 'Machine Learning Manager': 1,
                 'Data Engineering Manager': 5,
                 'Machine Learning Infrastructure Engineer': 3,
                 'ML Engineer': 6,
                 'AI Scientist': 7,
                 'Computer Vision Engineer': 6,
                 'Principal Data Scientist': 7,
                 'Data Science Manager': 12,
                 'Head of Data': 5,
                 '3D Computer Vision Researcher': 1,
                 'Data Analytics Engineer': 4,
                 'Applied Data Scientist': 5,
                 'Marketing Data Analyst': 1,
                 'Cloud Data Engineer': 2,
                 'Financial Data Analyst': 2,
                 'Computer Vision Software Engineer': 3,
                 'Director of Data Engineering': 2,
                 'Data Science Engineer': 3,
                 'Principal Data Engineer': 3,
                 'Machine Learning Developer': 3,
                 'Applied Machine Learning Scientist': 4,
                 'Data Analytics Manager': 7,
                 'Head of Data Science': 4,
                 'Data Specialist': 1,
                 'Data Architect': 11,
                 'Finance Data Analyst': 1,
                 'Principal Data Analyst': 2,
                 'Big Data Architect': 1,
                 'Staff Data Scientist': 1,
                 'Analytics Engineer': 4,
                 'ETL Developer': 2,
                 'Head of Machine Learning': 1,
                 'NLP Engineer': 1,
                 'Lead Machine Learning Engineer': 1,
                 'Data Analytics Lead': 1})
```

b.3) Países con empleados residentes en el extranjero

In [14]:

```
# Esta celda debe ser completada por el estudiante
def paises_con_empleados_en_extranjero_anno_dado(input_file_path: str, year: int):
    """
    Reads a CSV file and extracts information about job positions, country pairs, a

    This function reads the specified CSV file, filters the data based on the given
    about job positions, country pairs (if the source and destination countries are

    Parameters
    ----------
    input_file_path : str
        The path to the input CSV file.

    year : int
        The year to filter the data.

    Returns
    -------
    tuple
        A tuple containing:
        - employments: A set of unique job positions for the specified year.
        - country_pairs: A dictionary with pairs of source and destination countrie
    """
    # Initialize sets and dictionaries to store data
    country_pairs = {}

    try:
        with open(input_file_path, 'r', newline='') as infile:
            reader = infile.readlines()
            for line in reader:
                row = line.strip().split(';')
                if row[0] == str(year):
                    if row[5] != row[7]:  # Check if source and destination countri
                        pair = (row[5], row[7])
                        if pair in country_pairs:
                            country_pairs[pair] += 1
                        else:
                            country_pairs[pair] = 1

        return country_pairs
    except FileNotFoundError:
        print(f"File '{input_file_path}' not found.")
    except Exception as e:
        print(f"An error occurred: {str(e)}")
```

In [15]:

```
# Comprobación de funcionamiento:

paises_con_empleados_en_extranjero_anno_dado(DatosSalariosNormalizados, 2021)
```

Out[15]:
```
{('IN', 'US'): 3,
 ('GB', 'CA'): 1,
 ('IT', 'PL'): 1,
 ('BG', 'US'): 1,
 ('GR', 'DK'): 1,
 ('BR', 'US'): 2,
 ('DE', 'US'): 1,
 ('HU', 'US'): 1,
 ('PK', 'US'): 1,
 ('ES', 'RO'): 1,
 ('VN', 'US'): 1,
 ('SG', 'IL'): 1,
 ('RO', 'US'): 1,
 ('VN', 'GB'): 1,
 ('FR', 'ES'): 1,
 ('RO', 'GB'): 1,
 ('US', 'FR'): 1,
 ('DE', 'AT'): 1,
 ('FR', 'US'): 1,
 ('IT', 'US'): 1,
 ('HK', 'GB'): 1,
 ('IN', 'CH'): 1,
 ('US', 'CA'): 1,
 ('IN', 'AS'): 1,
 ('RS', 'DE'): 1,
 ('PR', 'US'): 1,
 ('NL', 'DE'): 1,
 ('JE', 'CN'): 1}
```

b.4) Ídem, usando diccionarios por defecto

In [16]:
```python
# Esta celda debe ser completada por el estudiante
def anno_cargos_paises_comps_empls(input_file_path: str, year: int):
    """
    Reads a CSV file and extracts information about job positions, country pairs, a

    This function reads the specified CSV file, filters the data based on the given
    about job positions, country pairs (if the source and destination countries are

    Parameters
    ----------
    input_file_path : str
        The path to the input CSV file.

    year : int
        The year to filter the data.

    Returns
    -------
    tuple
        A tuple containing:
        - employments: A set of unique job positions for the specified year.
        - country_pairs: A defaultdict with pairs of source and destination countri
    """
```

```python
    # job frequences
    country_pairs = defaultdict(int)
    try:
        with open(input_file_path, 'r', newline='') as infile:
            reader = infile.readlines()
            for line in reader:
                row = line.strip().split(';')
                if row[0] == str(year):
                    if row[5] != row[7]:
                        if(row[5], row[7]) in country_pairs:
                            country_pairs[(row[5], row[7])] = country_pairs[(row[5]
                        else:
                            country_pairs[(row[5], row[7])] = 1
        return country_pairs
    except FileNotFoundError:
        print(f"File '{input_file_path}' not found.")
    except Exception as e:
        print(f"An error occurred: {str(e)}")
```

In [17]:
```python
anno_cargos_paises_comps_empls(DatosSalariosNormalizados, 2021)
```

Out[17]:
```
defaultdict(int,
            {('IN', 'US'): 3,
             ('GB', 'CA'): 1,
             ('IT', 'PL'): 1,
             ('BG', 'US'): 1,
             ('GR', 'DK'): 1,
             ('BR', 'US'): 2,
             ('DE', 'US'): 1,
             ('HU', 'US'): 1,
             ('PK', 'US'): 1,
             ('ES', 'RO'): 1,
             ('VN', 'US'): 1,
             ('SG', 'IL'): 1,
             ('RO', 'US'): 1,
             ('VN', 'GB'): 1,
             ('FR', 'ES'): 1,
             ('RO', 'GB'): 1,
             ('US', 'FR'): 1,
             ('DE', 'AT'): 1,
             ('FR', 'US'): 1,
             ('IT', 'US'): 1,
             ('HK', 'GB'): 1,
             ('IN', 'CH'): 1,
             ('US', 'CA'): 1,
             ('IN', 'AS'): 1,
             ('RS', 'DE'): 1,
             ('PR', 'US'): 1,
             ('NL', 'DE'): 1,
             ('JE', 'CN'): 1})
```

En este apartado, aunque se han tenido algunos problemas lógicos a la hora de iterar sobre las estructuras de datos manejadas, finalmente se han completado con éxito todos los enunciados propuestos.

c) Un diccionario se parece a una tabla... [1'5 puntos]

c.1) Carga de los datos en una tabla (compacta)

Para cada tipo de puesto, nivel, año y país, deseamos tener la relación de salarios. Cargaremos esta información en un diccionario cuyas claves serán tuplas (con los puestos, el nivel, el año y el país) y cuyo valor será la relación de salarios. La idea es que podamos luego acceder a la información de la siguiente manera:

```
('Data Scientist', 3, 2020, 'US'):
[68428, 120000, 412000]
```

Te pido una versión de lectura de los datos en una tabla como ésta.

In [18]:
```python
# Esta celda debe ser completada por el estudiante

def load_salaries_compact(input_file_path: str) -> Dict[Tuple[str, int, int, str],
    """
    Reads a CSV file and extracts information about job positions and corresponding

    This function reads the specified CSV file, processes the data, and organizes i
    tuple containing job position, year, month, and country, and the value is a lis
    key.

    Parameters
    ----------
    input_file_path : str
        The path to the input CSV file.

    Returns
    -------
    Dict[Tuple[str, int, int, str], List[int]]
        A dictionary where each key is a tuple with job position, year, month, and
        corresponding salaries.

    Raises
    ------
    FileNotFoundError
        If the specified input file is not found.

    Returns
    -------
    Dict[Tuple[str, int, int, str], List[int]]
        A dictionary containing job positions as keys and lists of associated salar
    """
    # job frequences
    compact_dict = defaultdict(list)
    try:
```

```python
        with open(input_file_path, 'r', newline='') as infile:
            reader = infile.readlines()
            for line in reader:
                row = line.strip().split(';')
                key = (row[3], int(row[1]), int(row[0]), row[7])
                if key in compact_dict:
                    compact_dict[key].append(int(row[4]))
                else:
                    compact_dict[key] = [int(row[4])]
        return compact_dict
    except FileNotFoundError:
        print(f"File '{input_file_path}' not found.")
    except Exception as e:
        print(f"An error occurred: {str(e)}")
```

In [19]:

```python
# Comprobación:

Salarios_tabla_compact = load_salaries_compact(DatosSalariosNormalizados)
print(Salarios_tabla_compact)
```

```
defaultdict(<class 'list'>, {('Data Scientist', 1, 2020, 'DE'): [79833], ('Machine L
earning Scientist', 3, 2020, 'JP'): [260000], ('Big Data Engineer', 3, 2020, 'GB'):
[109024, 114047], ('Product Data Analyst', 1, 2020, 'HN'): [20000], ('Machine Learni
ng Engineer', 3, 2020, 'US'): [150000], ('Data Analyst', 0, 2020, 'US'): [72000, 910
00], ('Lead Data Scientist', 3, 2020, 'US'): [190000], ('Data Scientist', 1, 2020,
'HU'): [35735], ('Business Data Analyst', 1, 2020, 'US'): [135000], ('Lead Data Engi
neer', 3, 2020, 'NZ'): [125000], ('Data Scientist', 0, 2020, 'FR'): [51321, 39916],
('Data Scientist', 1, 2020, 'IN'): [40481], ('Lead Data Analyst', 1, 2020, 'US'): [8
7000], ('Data Analyst', 1, 2020, 'US'): [85000], ('Data Analyst', 1, 2020, 'PK'): [8
000], ('Data Engineer', 0, 2020, 'JP'): [41689], ('Data Science Consultant', 0, 202
0, 'IN'): [5707], ('Lead Data Engineer', 1, 2020, 'US'): [56000], ('Machine Learning
Engineer', 1, 2020, 'CN'): [43331], ('Product Data Analyst', 1, 2020, 'IN'): [6072],
('Data Engineer', 3, 2020, 'GR'): [47899], ('BI Data Analyst', 1, 2020, 'US'): [9800
0], ('Lead Data Scientist', 1, 2020, 'AE'): [115000], ('Director of Data Science',
2, 2020, 'US'): [325000], ('Research Scientist', 0, 2020, 'NL'): [42000], ('Data Eng
ineer', 3, 2020, 'MX'): [33511], ('Business Data Analyst', 0, 2020, 'US'): [100000],
('Machine Learning Manager', 3, 2020, 'CA'): [117104], ('Data Engineering Manager',
1, 2020, 'DE'): [59303], ('Big Data Engineer', 0, 2020, 'US'): [70000], ('Data Scien
tist', 3, 2020, 'US'): [68428, 120000, 412000], ('Research Scientist', 1, 2020, 'U
S'): [450000], ('Data Analyst', 1, 2020, 'FR'): [46759], ('Data Engineer', 1, 2020,
'AT'): [74130], ('Data Science Consultant', 1, 2020, 'US'): [103000], ('Machine Lear
ning Engineer', 0, 2020, 'US'): [250000, 138000], ('Data Analyst', 0, 2020, 'NG'):
[10000], ('Data Scientist', 1, 2020, 'US'): [45760, 105000, 118000, 138350], ('Data
Engineering Manager', 2, 2020, 'ES'): [79833], ('Machine Learning Infrastructure Eng
ineer', 1, 2020, 'PT'): [50180], ('Data Engineer', 1, 2020, 'US'): [106000, 110000,
130800], ('Data Engineer', 1, 2020, 'GB'): [112872], ('ML Engineer', 0, 2020, 'DE'):
[15966], ('Data Scientist', 1, 2020, 'GB'): [76958], ('Data Engineer', 3, 2020, 'U
S'): [188000], ('Data Engineer', 1, 2020, 'FR'): [70139], ('Data Analyst', 0, 2020,
'IN'): [6072], ('AI Scientist', 0, 2020, 'DK'): [45896], ('Data Engineer', 0, 2020,
'DE'): [54742], ('Computer Vision Engineer', 3, 2020, 'US'): [60000], ('Principal Da
ta Scientist', 3, 2020, 'DE'): [148261], ('Data Scientist', 1, 2020, 'ES'): [38776],
('Data Scientist', 0, 2020, 'IT'): [21669], ('Machine Learning Engineer', 3, 2020,
'HR'): [45618], ('Data Scientist', 0, 2020, 'DE'): [62726, 49268], ('Data Science Ma
nager', 3, 2020, 'US'): [190200], ('Data Scientist', 0, 2020, 'US'): [105000], ('Dat
a Scientist', 3, 2020, 'AT'): [91237], ('Data Scientist', 1, 2020, 'LU'): [62726],
('Data Scientist', 1, 2020, 'FR'): [42197], ('Research Scientist', 0, 2021, 'GB'):
[82528], ('BI Data Analyst', 2, 2021, 'US'): [150000], ('Head of Data', 2, 2021, 'U
S'): [235000], ('Data Scientist', 3, 2021, 'FR'): [53192, 77684], ('BI Data Analys
t', 1, 2021, 'US'): [100000, 36259], ('3D Computer Vision Researcher', 1, 2021, 'I
N'): [5409], ('ML Engineer', 1, 2021, 'US'): [270000], ('Data Analyst', 0, 2021, 'U
S'): [80000, 90000, 50000, 60000], ('Data Analytics Engineer', 3, 2021, 'DE'): [7919
7], ('Data Engineer', 1, 2021, 'US'): [140000, 200000, 100000, 90000, 26005, 20000,
110000, 200000, 93150, 111775, 112000], ('Applied Data Scientist', 1, 2021, 'CA'):
[54238], ('Machine Learning Engineer', 1, 2021, 'ES'): [47282], ('Director of Data S
cience', 2, 2021, 'PL'): [153667], ('Data Engineer', 1, 2021, 'PL'): [28476], ('Data
Analyst', 0, 2021, 'FR'): [59102], ('Data Analytics Engineer', 1, 2021, 'US'): [1100
00], ('Lead Data Analyst', 3, 2021, 'US'): [170000], ('Data Analyst', 3, 2021, 'U
S'): [80000, 200000], ('Marketing Data Analyst', 3, 2021, 'DK'): [88654], ('Data Sci
ence Consultant', 0, 2021, 'DE'): [76833, 63831, 76833], ('Lead Data Analyst', 1, 20
21, 'IN'): [19609], ('Lead Data Engineer', 3, 2021, 'US'): [276000, 160000], ('Data
Scientist', 0, 2021, 'IN'): [29751, 28399], ('Cloud Data Engineer', 1, 2021, 'SG'):
[89294], ('AI Scientist', 0, 2021, 'US'): [12000, 12000], ('Financial Data Analyst',
1, 2021, 'US'): [450000], ('Computer Vision Software Engineer', 0, 2021, 'US'): [700
00], ('Computer Vision Software Engineer', 1, 2021, 'US'): [95746], ('Data Analyst',
1, 2021, 'US'): [75000, 62000, 90000, 135000, 80000, 93000], ('Data Engineer', 3, 20
21, 'US'): [150000, 115000, 150000, 165000], ('Data Scientist', 1, 2021, 'US'): [730
```

00, 82500, 150000, 5679, 147000, 160000, 115000, 130000, 109000], ('Data Analyst', 1, 2021, 'GB'): [51519], ('Research Scientist', 1, 2021, 'CA'): [187442, 63810], ('Data Engineer', 0, 2021, 'IN'): [30428, 21637], ('Machine Learning Engineer', 3, 2021, 'DE'): [94564], ('Director of Data Engineering', 3, 2021, 'GB'): [113476], ('Lead Data Engineer', 3, 2021, 'GB'): [103160], ('Data Engineer', 1, 2021, 'NL'): [45391, 69741], ('Machine Learning Scientist', 0, 2021, 'US'): [225000], ('Data Scientist', 1, 2021, 'NG'): [50000], ('Data Science Engineer', 1, 2021, 'GR'): [40189], ('Big Data Engineer', 1, 2021, 'RO'): [60000], ('Principal Data Engineer', 3, 2021, 'US'): [200000, 185000], ('Applied Data Scientist', 0, 2021, 'GB'): [110037], ('Data Analyst', 0, 2021, 'ES'): [10354], ('Principal Data Scientist', 1, 2021, 'US'): [151000], ('Machine Learning Scientist', 3, 2021, 'US'): [120000], ('Data Scientist', 1, 2021, 'IN'): [9466, 33808, 16904], ('Machine Learning Engineer', 0, 2021, 'IN'): [20000], ('Lead Data Scientist', 3, 2021, 'IN'): [40570], ('Machine Learning Developer', 0, 2021, 'IQ'): [100000], ('Data Scientist', 0, 2021, 'FR'): [49646, 36643], ('Applied Machine Learning Scientist', 1, 2021, 'US'): [38400, 423000], ('Computer Vision Engineer', 3, 2021, 'BR'): [24000, 18907], ('Data Scientist', 0, 2021, 'US'): [100000, 80000, 90000, 100000, 58000], ('ML Engineer', 1, 2021, 'JP'): [63711, 77364], ('Principal Data Scientist', 3, 2021, 'US'): [220000, 235000], ('Data Science Manager', 3, 2021, 'US'): [240000, 144000, 174000, 54094], ('Data Engineering Manager', 3, 2021, 'US'): [150000, 153000, 174000], ('Machine Learning Engineer', 3, 2021, 'BE'): [82744], ('Research Scientist', 1, 2021, 'FR'): [62649, 56738], ('Cloud Data Engineer', 3, 2021, 'US'): [160000], ('Director of Data Science', 3, 2021, 'JP'): [168000], ('Data Scientist', 1, 2021, 'CA'): [75774], ('Data Scientist', 0, 2021, 'UA'): [13400], ('Data Science Engineer', 3, 2021, 'CA'): [127221], ('Data Scientist', 1, 2021, 'IL'): [119059], ('Data Analytics Manager', 3, 2021, 'US'): [120000, 120000, 140000], ('Machine Learning Engineer', 0, 2021, 'US'): [125000, 81000], ('Head of Data', 2, 2021, 'RU'): [230000], ('Head of Data Science', 2, 2021, 'RU'): [85000], ('Data Engineer', 1, 2021, 'MT'): [28369], ('Director of Data Science', 2, 2021, 'DE'): [130026, 141846], ('Data Specialist', 3, 2021, 'US'): [165000], ('Data Engineer', 0, 2021, 'US'): [80000, 72500], ('Director of Data Science', 2, 2021, 'US'): [250000], ('BI Data Analyst', 0, 2021, 'US'): [55000], ('Data Architect', 1, 2021, 'US'): [150000, 170000, 180000], ('Data Engineer', 1, 2021, 'GB'): [82528, 66022, 72212], ('Research Scientist', 3, 2021, 'PT'): [60757], ('Data Scientist', 1, 2021, 'MX'): [2859], ('Data Scientist', 1, 2021, 'CL'): [40038], ('Big Data Engineer', 1, 2021, 'IN'): [22611], ('Data Scientist', 1, 2021, 'DE'): [90734, 90734, 88654, 25532], ('Finance Data Analyst', 3, 2021, 'GB'): [61896], ('Machine Learning Scientist', 1, 2021, 'PK'): [12000], ('Data Engineer', 1, 2021, 'IR'): [4000], ('Data Analytics Engineer', 3, 2021, 'GB'): [50000], ('Data Science Consultant', 2, 2021, 'ES'): [69741], ('Data Engineer', 3, 2021, 'GB'): [76833, 96282], ('Machine Learning Engineer', 1, 2021, 'JP'): [74000], ('Data Science Manager', 3, 2021, 'FR'): [152000], ('Machine Learning Engineer', 0, 2021, 'CO'): [21844], ('Big Data Engineer', 1, 2021, 'MD'): [18000], ('Research Scientist', 3, 2021, 'CA'): [96113], ('BI Data Analyst', 0, 2021, 'KE'): [9272], ('Machine Learning Engineer', 3, 2021, 'IN'): [24342, 66265], ('Data Science Consultant', 0, 2021, 'US'): [90000], ('Data Scientist', 1, 2021, 'AT'): [61467], ('Machine Learning Infrastructure Engineer', 3, 2021, 'US'): [195000], ('Data Scientist', 1, 2021, 'ES'): [37825, 46809], ('Research Scientist', 3, 2021, 'US'): [50000], ('Data Scientist', 1, 2021, 'BR'): [12901], ('Machine Learning Engineer', 3, 2021, 'US'): [200000, 185000], ('Machine Learning Engineer', 1, 2021, 'SI'): [24823], ('Big Data Engineer', 0, 2021, 'CH'): [5882], ('Machine Learning Engineer', 0, 2021, 'DE'): [24823, 85000], ('Computer Vision Engineer', 0, 2021, 'DK'): [28609], ('Machine Learning Engineer', 1, 2021, 'BE'): [88654], ('Machine Learning Engineer', 1, 2021, 'PL'): [46597], ('Data Scientist', 1, 2021, 'GB'): [116914, 56256], ('Machine Learning Scientist', 3, 2021, 'CA'): [225000], ('Principal Data Scientist', 2, 2021, 'US'): [416000], ('Data Scientist', 3, 2021, 'CA'): [87738, 103691], ('Data Scientist', 3, 2021, 'US'): [135000, 165000], ('Data Analyst', 3, 2021, 'CA'): [71786], ('Big Data Engineer', 0, 2021, 'IN'): [16228], ('ML Engineer', 3, 2021, 'US'): [256000], ('Director o

f Data Engineering', 3, 2021, 'US'): [200000], ('Head of Data Science', 1, 2021, 'U
S'): [110000], ('Data Scientist', 0, 2021, 'VN'): [4000], ('AI Scientist', 0, 2021,
'AS'): [18053], ('Data Engineer', 1, 2021, 'TR'): [12103, 28016], ('Principal Data A
nalyst', 3, 2021, 'US'): [170000], ('Principal Data Engineer', 2, 2021, 'US'): [6000
00], ('Big Data Architect', 3, 2021, 'CA'): [99703], ('Principal Data Scientist', 3,
2021, 'DE'): [173762], ('Data Analyst', 3, 2021, 'DE'): [63831], ('Data Engineer',
0, 2021, 'DE'): [65013], ('AI Scientist', 3, 2021, 'ES'): [55000], ('Data Scientis
t', 3, 2021, 'TR'): [20171], ('Business Data Analyst', 0, 2021, 'LU'): [59102], ('Re
search Scientist', 0, 2021, 'CN'): [100000], ('Staff Data Scientist', 3, 2021, 'U
S'): [105000], ('Research Scientist', 1, 2021, 'CZ'): [69999], ('Data Science Manage
r', 3, 2021, 'IN'): [94665], ('Head of Data', 3, 2021, 'SI'): [102839], ('Machine Le
arning Engineer', 1, 2021, 'IT'): [51064], ('Data Engineer', 3, 2022, 'US'): [13500
0, 181940, 132320, 220110, 160080, 165400, 132320, 243900, 128875, 93700, 156600, 10
8800, 113000, 160000, 136000, 165400, 136994, 101570, 132320, 155000, 209100, 15460
0, 175000, 183600, 100800, 209100, 154600, 180000, 80000, 105000, 100000, 210000, 11
5000, 155000, 130000, 115000, 110500, 130000, 160000, 200100, 160000, 145000, 70500,
175100, 140250, 54000, 100000, 25000, 220110, 160080, 154000, 126000], ('Data Analys
t', 3, 2022, 'US'): [155000, 120600, 102100, 84900, 99000, 116000, 90320, 124190, 17
0000, 112900, 90320, 112900, 90320, 136600, 109280, 135000, 132000, 128875, 93700, 1
64000, 112900, 90320, 115934, 81666, 135000, 100000, 90320, 112900, 115934, 81666, 9
9050, 116150, 170000, 80000, 100000, 69000, 150075, 126500, 106260, 105000, 110925,
99000, 60000, 170000, 129000, 150000], ('Data Scientist', 1, 2022, 'US'): [130000, 9
0000, 200000, 120000, 100000, 48000, 135000, 78000, 141300, 102100, 160000, 130000],
('Data Engineer', 1, 2022, 'US'): [170000, 150000, 63900, 82900, 206699, 99100], ('D
ata Scientist', 3, 2022, 'US'): [136620, 99360, 146000, 123000, 165220, 120160, 1800
00, 120000, 95550, 167000, 123000, 150000, 211500, 138600, 170000, 123000, 215300, 1
58200, 180000, 260000, 180000, 80000, 140400, 215300, 104890, 140000, 220000, 14000
0, 185100, 230000, 100000, 165000, 205300, 140400, 176000, 144000, 205300, 140400, 1
40000, 210000, 140000, 210000, 140000, 210000, 140000, 230000, 150000, 210000], ('Da
ta Scientist', 3, 2022, 'GB'): [117789, 104702], ('Data Engineer', 0, 2022, 'GB'):
[52351, 45807], ('Data Analyst', 1, 2022, 'US'): [106260, 126500, 115500, 167000, 58
000, 58000, 135000, 50000], ('Data Engineer', 2, 2022, 'US'): [242000, 200000, 32400
0, 216000], ('Data Scientist', 1, 2022, 'GB'): [65438, 39263, 71982, 45807, 183228,
91614], ('Data Engineer', 1, 2022, 'GB'): [78526, 52351, 117789, 98158, 78526, 5889
4, 104702, 91614, 98158, 78526], ('Data Analyst', 2, 2022, 'US'): [130000, 110000],
('Head of Data Science', 2, 2022, 'US'): [224000, 167875], ('Analytics Engineer', 2,
2022, 'US'): [175000, 135000], ('Data Science Manager', 3, 2022, 'US'): [161342, 137
141, 152500], ('Data Engineer', 3, 2022, 'GB'): [78526, 65438], ('Data Architect',
3, 2022, 'CA'): [192400, 90700], ('Data Analyst', 3, 2022, 'CA'): [130000, 61300, 13
0000, 61300], ('Machine Learning Engineer', 3, 2022, 'US'): [189650, 164996, 189650,
164996, 220000, 120000, 214000, 192600], ('ETL Developer', 1, 2022, 'GR'): [54957, 5
4957], ('Lead Data Engineer', 2, 2022, 'CA'): [118187], ('Data Architect', 3, 2022,
'US'): [208775, 147800, 266400, 213120, 192564, 144854], ('Head of Machine Learnin
g', 2, 2022, 'IN'): [79039], ('Machine Learning Engineer', 0, 2022, 'GB'): [37300],
('Machine Learning Engineer', 1, 2022, 'GB'): [124333, 98158], ('AI Scientist', 1, 2
022, 'US'): [120000, 200000], ('Data Analytics Manager', 3, 2022, 'US'): [145000, 10
5400, 150260, 109280], ('Machine Learning Engineer', 1, 2022, 'DE'): [87932], ('Data
Analyst', 1, 2022, 'GB'): [52351, 39263, 65438, 45807], ('Data Engineer', 1, 2022,
'GR'): [65949, 49461, 87932, 76940], ('Data Science Engineer', 3, 2022, 'MX'): [6000
0], ('Machine Learning Scientist', 1, 2022, 'US'): [160000, 112300, 153000], ('Data
Science Manager', 1, 2022, 'US'): [241000, 159000], ('Data Engineer', 1, 2022, 'E
S'): [49461, 87932, 76940, 65949], ('Data Analyst', 1, 2022, 'ES'): [43966, 32974],
('Data Analyst', 1, 2022, 'GR'): [43966, 32974, 20000], ('ML Engineer', 0, 2022, 'P
T'): [21983], ('Machine Learning Developer', 1, 2022, 'CA'): [78791], ('Director of
Data Science', 2, 2022, 'CA'): [196979], ('Machine Learning Engineer', 1, 2022, 'U
S'): [120000], ('Computer Vision Engineer', 0, 2022, 'US'): [125000], ('NLP Enginee

```
r', 1, 2022, 'US'): [37236], ('Lead Machine Learning Engineer', 3, 2022, 'DE'): [879
32], ('Business Data Analyst', 1, 2022, 'IN'): [18442], ('Data Scientist', 1, 2022,
'IN'): [31615], ('Machine Learning Infrastructure Engineer', 1, 2022, 'PT'): [5825
5], ('Financial Data Analyst', 0, 2022, 'US'): [100000], ('Data Engineer', 1, 2022,
'DE'): [54957], ('Data Scientist', 0, 2022, 'IN'): [18442], ('Principal Data Scienti
st', 3, 2022, 'DE'): [162674], ('Data Engineer', 0, 2022, 'US'): [120000, 65000],
('Research Scientist', 3, 2022, 'US'): [144000], ('Machine Learning Engineer', 3, 20
22, 'AE'): [120000, 65000], ('Data Scientist', 0, 2022, 'DZ'): [100000], ('Applied M
achine Learning Scientist', 0, 2022, 'CZ'): [31875], ('Head of Data', 3, 2022, 'U
S'): [200000], ('Principal Data Analyst', 1, 2022, 'CA'): [75000], ('Data Scientis
t', 1, 2022, 'PL'): [35590], ('Machine Learning Developer', 3, 2022, 'CA'): [78791],
('Data Engineer', 0, 2022, 'DE'): [58035], ('Research Scientist', 3, 2022, 'FR'): [9
3427], ('Data Scientist', 0, 2022, 'CA'): [52396], ('Machine Learning Engineer', 3,
2022, 'NL'): [62651], ('Head of Data', 1, 2022, 'EE'): [32974], ('Data Scientist',
0, 2022, 'MY'): [40000], ('Machine Learning Engineer', 1, 2022, 'AU'): [87425], ('Da
ta Scientist', 0, 2022, 'AU'): [86703], ('Applied Machine Learning Scientist', 1, 20
22, 'US'): [75000], ('Research Scientist', 1, 2022, 'AT'): [64849], ('Research Scien
tist', 0, 2022, 'US'): [120000], ('Applied Data Scientist', 1, 2022, 'US'): [15700
0], ('Computer Vision Software Engineer', 0, 2022, 'AU'): [150000], ('Business Data
Analyst', 1, 2022, 'CA'): [70912], ('Machine Learning Engineer', 3, 2022, 'IE'): [71
444], ('Data Analytics Engineer', 0, 2022, 'PK'): [20000], ('Data Engineer', 1, 202
2, 'FR'): [68147], ('Data Scientist', 1, 2022, 'CH'): [122346], ('Applied Data Scien
tist', 3, 2022, 'US'): [380000, 177000], ('Data Scientist', 1, 2022, 'CA'): [69336],
('Computer Vision Engineer', 0, 2022, 'LU'): [10000], ('Data Analytics Lead', 3, 202
2, 'US'): [405000], ('Data Analyst', 1, 2022, 'CA'): [85000, 75000], ('Analytics Eng
ineer', 3, 2022, 'US'): [205300, 184700], ('Data Analyst', 0, 2022, 'CA'): [67000, 5
2000]})
```

c.2) Carga de todos los datos en una tabla de tablas...

Para cada tipo de puesto, año y país, deseamos tener la relación de salarios. En esta segunda versión, cargaremos esta información en un diccionario cuyas claves serán los puestos y cuyo valor, un nuevo diccionario con el año como clave y cuyo valor será un diccionario con el país como clave y la relación de salarios como valor. Aunque esto parece algo lioso, la idea es que podamos luego acceder a la información de la siguiente manera:

```
    Salarios["Data Scientist"][2021]["US"]
    [73000, 100000, 80000, 82500, 150000, 147000, 160000, 135000,
165000, 115000, 90000, 130000, 100000, 58000, 109000]
```

In [20]:
```python
# Esta celda debe ser completada por el estudiante
def load_salaries(input_file_path: str):
    """
    Reads a CSV file and extracts information about job positions, country pairs, a

    This function reads the specified CSV file, filters the data based on the given
    about job positions, country pairs (if the source and destination countries are

    Parameters
    ----------
    input_file_path : str
```

```
        The path to the input CSV file.

    Returns
    -------
    dict
        A dictionary containing the following structure:
        {
            source_country: {
                year: {
                    destination_country: [list of employment frequencies]
                }
            }
        }

    This function reads the CSV file specified by 'input_file_path' and structures
    where the keys represent the source country, year, and destination country. It

    If the file does not exist, a FileNotFoundError is raised and an error message
    during the file processing, an error message is printed as well.
    """
    # job frequences
    table_of_tables = defaultdict(lambda: {})
    try:
        with open(input_file_path, 'r') as infile:
            reader = infile.readlines()
            for line in reader:
                row = line.strip().split(';')
                if not row[3] in table_of_tables:
                    table_of_tables[row[3]] = defaultdict(lambda: {})
                if not int(row[0]) in table_of_tables[row[3]]:
                     table_of_tables[row[3]][int(row[0])] = defaultdict(lambda: {})
                if not row[7] in table_of_tables[row[3]] [int(row[0])]:
                    table_of_tables[row[3]] [int(row[0])][row[7]] = [int(row[4])]
                else:
                    table_of_tables[row[3]] [int(row[0])][row[7]].append(int(row[4]
        return table_of_tables
    except FileNotFoundError:
        print(f"File '{input_file_path}' not found.")
    except Exception as e:
        print(f"An error occurred: {str(e)}")
```

In [21]:

```
# Comprobación de funcionamiento, con los estados de Florida y Texas:

Salarios = load_salaries(DatosSalariosNormalizados)
print(Salarios)
```

```
defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839A67920>, {'Data
Scientist': defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B832EC3
380>, {2020: defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839A6
7A60>, {'DE': [79833, 62726, 49268], 'HU': [35735], 'FR': [51321, 39916, 42197], 'I
N': [40481], 'US': [68428, 45760, 105000, 118000, 120000, 138350, 412000, 105000],
'GB': [76958], 'ES': [38776], 'IT': [21669], 'AT': [91237], 'LU': [62726]}), 2021: d
efaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD68E0>, {'FR':
[53192, 49646, 36643, 77684], 'IN': [29751, 9466, 33808, 28399, 16904], 'US': [7300
0, 100000, 80000, 82500, 150000, 5679, 147000, 160000, 135000, 165000, 115000, 9000
0, 130000, 100000, 58000, 109000], 'NG': [50000], 'CA': [75774, 87738, 103691], 'U
A': [13400], 'IL': [119059], 'MX': [2859], 'CL': [40038], 'DE': [90734, 90734, 8865
4, 25532], 'AT': [61467], 'ES': [37825, 46809], 'BR': [12901], 'GB': [116914, 5625
6], 'VN': [4000], 'TR': [20171]}), 2022: defaultdict(<function load_salaries.<locals
>.<lambda> at 0x000002B839AD9BC0>, {'US': [130000, 90000, 136620, 99360, 146000, 123
000, 165220, 120160, 180000, 120000, 95550, 167000, 123000, 150000, 211500, 138600,
170000, 123000, 215300, 158200, 180000, 260000, 180000, 80000, 140400, 215300, 10489
0, 140000, 220000, 140000, 185100, 200000, 120000, 230000, 100000, 100000, 165000, 4
8000, 135000, 78000, 141300, 102100, 205300, 140400, 176000, 144000, 205300, 140400,
140000, 210000, 140000, 210000, 140000, 210000, 140000, 230000, 150000, 210000, 1600
00, 130000], 'GB': [117789, 104702, 65438, 39263, 71982, 45807, 183228, 91614], 'I
N': [31615, 18442], 'DZ': [100000], 'PL': [35590], 'CA': [52396, 69336], 'MY': [4000
0], 'AU': [86703], 'CH': [122346]})}), 'Machine Learning Scientist': defaultdict(<fu
nction load_salaries.<locals>.<lambda> at 0x000002B839A67C40>, {2020: defaultdict(<f
unction load_salaries.<locals>.<lambda> at 0x000002B839A67CE0>, {'JP': [260000]}), 2
021: defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD7E20>,
{'US': [225000, 120000], 'PK': [12000], 'CA': [225000]}), 2022: defaultdict(<functio
n load_salaries.<locals>.<lambda> at 0x000002B839ADA700>, {'US': [160000, 112300, 15
3000]})}), 'Big Data Engineer': defaultdict(<function load_salaries.<locals>.<lambda
> at 0x000002B839A67E20>, {2020: defaultdict(<function load_salaries.<locals>.<lambd
a> at 0x000002B839A67EC0>, {'GB': [109024, 114047], 'US': [70000]}), 2021: defaultdi
ct(<function load_salaries.<locals>.<lambda> at 0x000002B839AD80E0>, {'RO': [60000],
'IN': [22611, 16228], 'MD': [18000], 'CH': [5882]})}), 'Product Data Analyst': defau
ltdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD4040>, {2020: defa
ultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD40E0>, {'HN': [20
000], 'IN': [6072]})}), 'Machine Learning Engineer': defaultdict(<function load_sala
ries.<locals>.<lambda> at 0x000002B839AD4220>, {2020: defaultdict(<function load_sal
aries.<locals>.<lambda> at 0x000002B839AD42C0>, {'US': [150000, 250000, 138000], 'C
N': [43331], 'HR': [45618]}), 2021: defaultdict(<function load_salaries.<locals>.<la
mbda> at 0x000002B839AD7100>, {'ES': [47282], 'DE': [94564, 24823, 85000], 'IN': [20
000, 24342, 66265], 'BE': [82744, 88654], 'US': [125000, 81000, 200000, 185000], 'J
P': [74000], 'CO': [21844], 'SI': [24823], 'PL': [46597], 'IT': [51064]}), 2022: def
aultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839ADA020>, {'US': [1
89650, 164996, 189650, 164996, 120000, 220000, 120000, 214000, 192600], 'GB': [3730
0, 124333, 98158], 'DE': [87932], 'AE': [120000, 65000], 'NL': [62651], 'AU': [8742
5], 'IE': [71444]})}), 'Data Analyst': defaultdict(<function load_salaries.<locals>.
<lambda> at 0x000002B839AD4400>, {2020: defaultdict(<function load_salaries.<locals
>.<lambda> at 0x000002B839AD44A0>, {'US': [72000, 85000, 91000], 'PK': [8000], 'FR':
[46759], 'NG': [10000], 'IN': [6072]}), 2021: defaultdict(<function load_salaries.<l
ocals>.<lambda> at 0x000002B839AD6C00>, {'US': [80000, 80000, 75000, 62000, 90000, 5
0000, 90000, 135000, 60000, 200000, 80000, 93000], 'FR': [59102], 'GB': [51519], 'E
S': [10354], 'CA': [71786], 'DE': [63831]}), 2022: defaultdict(<function load_salari
es.<locals>.<lambda> at 0x000002B839AD9B20>, {'US': [155000, 120600, 102100, 84900,
99000, 116000, 106260, 126500, 90320, 124190, 130000, 110000, 170000, 115500, 11290
0, 90320, 112900, 90320, 167000, 136600, 109280, 135000, 58000, 132000, 128875, 9370
0, 164000, 112900, 90320, 115934, 81666, 58000, 135000, 50000, 135000, 100000, 9032
0, 112900, 115934, 81666, 99050, 116150, 170000, 80000, 100000, 69000, 150075, 12650
```

```
0, 106260, 105000, 110925, 99000, 60000, 170000, 129000, 150000], 'CA': [130000, 613
00, 130000, 61300, 85000, 75000, 67000, 52000], 'GB': [52351, 39263, 65438, 45807],
'ES': [43966, 32974], 'GR': [43966, 32974, 20000]})}), 'Lead Data Scientist': defaul
tdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD45E0>, {2020: defau
ltdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD4680>, {'US': [190
000], 'AE': [115000]}), 2021: defaultdict(<function load_salaries.<locals>.<lambda>
at 0x000002B839AD8400>, {'IN': [40570]})}), 'Business Data Analyst': defaultdict(<fu
nction load_salaries.<locals>.<lambda> at 0x000002B839AD47C0>, {2020: defaultdict(<f
unction load_salaries.<locals>.<lambda> at 0x000002B839AD4860>, {'US': [135000, 1000
00]}), 2021: defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD
9800>, {'LU': [59102]}), 2022: defaultdict(<function load_salaries.<locals>.<lambda>
at 0x000002B839ADADE0>, {'IN': [18442], 'CA': [70912]})}), 'Lead Data Engineer': def
aultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD49A0>, {2020: de
faultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD4A40>, {'NZ':
[125000], 'US': [56000]}), 2021: defaultdict(<function load_salaries.<locals>.<lambd
a> at 0x000002B839AD7560>, {'US': [276000, 160000], 'GB': [103160]}), 2022: defaultd
ict(<function load_salaries.<locals>.<lambda> at 0x000002B839ADA2A0>, {'CA': [11818
7]})}), 'Lead Data Analyst': defaultdict(<function load_salaries.<locals>.<lambda> a
t 0x000002B839AD4B80>, {2020: defaultdict(<function load_salaries.<locals>.<lambda>
at 0x000002B839AD4C20>, {'US': [87000]}), 2021: defaultdict(<function load_salaries.
<locals>.<lambda> at 0x000002B839AD7240>, {'US': [170000], 'IN': [19609]})}), 'Data
Engineer': defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD4D
60>, {2020: defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD4
E00>, {'JP': [41689], 'GR': [47899], 'MX': [33511], 'AT': [74130], 'US': [106000, 18
8000, 110000, 130800], 'GB': [112872], 'FR': [70139], 'DE': [54742]}), 2021: default
dict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD6E80>, {'US': [14000
0, 150000, 115000, 150000, 200000, 100000, 90000, 80000, 26005, 165000, 20000, 11000
0, 200000, 93150, 111775, 72500, 112000], 'PL': [28476], 'IN': [30428, 21637], 'NL':
[45391, 69741], 'MT': [28369], 'GB': [82528, 76833, 66022, 72212, 96282], 'IR': [400
0], 'TR': [12103, 28016], 'DE': [65013]}), 2022: defaultdict(<function load_salarie
s.<locals>.<lambda> at 0x000002B839AD9A80>, {'US': [135000, 170000, 150000, 242000,
200000, 181940, 132320, 220110, 160080, 165400, 132320, 243900, 128875, 93700, 15660
0, 108800, 113000, 160000, 136000, 165400, 136994, 101570, 132320, 155000, 209100, 1
54600, 175000, 183600, 63900, 82900, 100800, 209100, 154600, 180000, 80000, 105000,
120000, 100000, 324000, 216000, 210000, 115000, 65000, 155000, 206699, 99100, 13000
0, 115000, 110500, 130000, 160000, 200100, 160000, 145000, 70500, 175100, 140250, 54
000, 100000, 25000, 220110, 160080, 154000, 126000], 'GB': [52351, 78526, 52351, 458
07, 78526, 65438, 117789, 98158, 78526, 58894, 104702, 91614, 98158, 78526], 'GR':
[65949, 49461, 87932, 76940], 'ES': [49461, 87932, 76940, 65949], 'DE': [54957, 5803
5], 'FR': [68147]})}), 'Data Science Consultant': defaultdict(<function load_salarie
s.<locals>.<lambda> at 0x000002B839AD4F40>, {2020: defaultdict(<function load_salari
es.<locals>.<lambda> at 0x000002B839AD4FE0>, {'IN': [5707], 'US': [103000]}), 2021:
defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD74C0>, {'DE':
[76833, 63831, 76833], 'ES': [69741], 'US': [90000]})}), 'BI Data Analyst': defaultd
ict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD5120>, {2020: default
dict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD51C0>, {'US': [9800
0]}), 2021: defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD6
660>, {'US': [150000, 100000, 36259, 55000], 'KE': [9272]})}), 'Director of Data Sci
ence': defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD5300>,
{2020: defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD53A0>,
{'US': [325000]}), 2021: defaultdict(<function load_salaries.<locals>.<lambda> at 0x
000002B839AD71A0>, {'PL': [153667], 'JP': [168000], 'DE': [130026, 141846], 'US': [2
50000]}), 2022: defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B83
9ADA8E0>, {'CA': [196979]})}), 'Research Scientist': defaultdict(<function load_sala
ries.<locals>.<lambda> at 0x000002B839AD54E0>, {2020: defaultdict(<function load_sal
aries.<locals>.<lambda> at 0x000002B839AD5580>, {'NL': [42000], 'US': [450000]}), 20
```

21: defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD65C0>,
{'GB': [82528], 'CA': [187442, 96113, 63810], 'FR': [62649, 56738], 'PT': [60757],
'US': [50000], 'CN': [100000], 'CZ': [69999]}), 2022: defaultdict(<function load_sal
aries.<locals>.<lambda> at 0x000002B839ADB060>, {'US': [144000, 120000], 'FR': [9342
7], 'AT': [64849]})}), 'Machine Learning Manager': defaultdict(<function load_salari
es.<locals>.<lambda> at 0x000002B839AD56C0>, {2020: defaultdict(<function load_salar
ies.<locals>.<lambda> at 0x000002B839AD5760>, {'CA': [117104]})}), 'Data Engineering
Manager': defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD58A
0>, {2020: defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD59
40>, {'DE': [59303], 'ES': [79833]}), 2021: defaultdict(<function load_salaries.<loc
als>.<lambda> at 0x000002B839AD89A0>, {'US': [150000, 153000, 174000]})}), 'Machine
Learning Infrastructure Engineer': defaultdict(<function load_salaries.<locals>.<lam
bda> at 0x000002B839AD5A80>, {2020: defaultdict(<function load_salaries.<locals>.<la
mbda> at 0x000002B839AD5B20>, {'PT': [50180]}), 2021: defaultdict(<function load_sal
aries.<locals>.<lambda> at 0x000002B839AD93A0>, {'US': [195000]}), 2022: defaultdict
(<function load_salaries.<locals>.<lambda> at 0x000002B839ADAE80>, {'PT': [5825
5]})}), 'ML Engineer': defaultdict(<function load_salaries.<locals>.<lambda> at 0x00
0002B839AD5C60>, {2020: defaultdict(<function load_salaries.<locals>.<lambda> at 0x0
00002B839AD5D00>, {'DE': [15966]}), 2021: defaultdict(<function load_salaries.<local
s>.<lambda> at 0x000002B839AD6B60>, {'US': [270000, 256000], 'JP': [63711, 77364]}),
2022: defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839ADA7A0>,
{'PT': [21983]})}), 'AI Scientist': defaultdict(<function load_salaries.<locals>.<la
mbda> at 0x000002B839AD5E40>, {2020: defaultdict(<function load_salaries.<locals>.<l
ambda> at 0x000002B839AD5EE0>, {'DK': [45896]}), 2021: defaultdict(<function load_sa
laries.<locals>.<lambda> at 0x000002B839AD77E0>, {'US': [12000, 12000], 'AS': [1805
3], 'ES': [55000]}), 2022: defaultdict(<function load_salaries.<locals>.<lambda> at
0x000002B839ADA520>, {'US': [120000, 200000]})}), 'Computer Vision Engineer': defaul
tdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD6020>, {2020: defau
ltdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD60C0>, {'US': [600
00]}), 2021: defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD
8860>, {'BR': [24000, 18907], 'DK': [28609]}), 2022: defaultdict(<function load_sala
ries.<locals>.<lambda> at 0x000002B839ADA980>, {'US': [125000], 'LU': [10000]})}),
'Principal Data Scientist': defaultdict(<function load_salaries.<locals>.<lambda> at
0x000002B839AD6200>, {2020: defaultdict(<function load_salaries.<locals>.<lambda> at
0x000002B839AD62A0>, {'DE': [148261]}), 2021: defaultdict(<function load_salaries.<l
ocals>.<lambda> at 0x000002B839AD8360>, {'US': [151000, 220000, 235000, 416000], 'D
E': [173762]}), 2022: defaultdict(<function load_salaries.<locals>.<lambda> at 0x000
002B839ADAFC0>, {'DE': [162674]})}), 'Data Science Manager': defaultdict(<function l
oad_salaries.<locals>.<lambda> at 0x000002B839AD63E0>, {2020: defaultdict(<function
load_salaries.<locals>.<lambda> at 0x000002B839AD6480>, {'US': [190200]}), 2021: def
aultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD8900>, {'US': [2
40000, 144000, 174000, 54094], 'FR': [152000], 'IN': [94665]}), 2022: defaultdict(<f
unction load_salaries.<locals>.<lambda> at 0x000002B839AD9EE0>, {'US': [161342, 1371
41, 241000, 159000, 152500]})}), 'Head of Data': defaultdict(<function load_salarie
s.<locals>.<lambda> at 0x000002B839AD6700>, {2021: defaultdict(<function load_salari
es.<locals>.<lambda> at 0x000002B839AD67A0>, {'US': [235000], 'RU': [230000], 'SI':
[102839]}), 2022: defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B
839ADB1A0>, {'US': [200000], 'EE': [32974]})}), '3D Computer Vision Researcher': def
aultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD6980>, {2021: de
faultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD6A20>, {'IN':
[5409]})}), 'Data Analytics Engineer': defaultdict(<function load_salaries.<locals>.
<lambda> at 0x000002B839AD6CA0>, {2021: defaultdict(<function load_salaries.<locals
>.<lambda> at 0x000002B839AD6D40>, {'DE': [79197], 'US': [110000], 'GB': [50000]}),
2022: defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839ADB420>,
{'PK': [20000]})}), 'Applied Data Scientist': defaultdict(<function load_salaries.<l
ocals>.<lambda> at 0x000002B839AD6F20>, {2021: defaultdict(<function load_salaries.<

```
locals>.<lambda> at 0x000002B839AD6FC0>, {'CA': [54238], 'GB': [110037]}), 2022: def
aultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839ADB2E0>, {'US': [1
57000, 380000, 177000]})}), 'Marketing Data Analyst': defaultdict(<function load_sal
aries.<locals>.<lambda> at 0x000002B839AD72E0>, {2021: defaultdict(<function load_sa
laries.<locals>.<lambda> at 0x000002B839AD7380>, {'DK': [88654]})}), 'Cloud Data Eng
ineer': defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD7600
>, {2021: defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD76A
0>, {'SG': [89294], 'US': [160000]})}), 'Financial Data Analyst': defaultdict(<funct
ion load_salaries.<locals>.<lambda> at 0x000002B839AD7880>, {2021: defaultdict(<func
tion load_salaries.<locals>.<lambda> at 0x000002B839AD7920>, {'US': [450000]}), 202
2: defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839ADAF20>, {'U
S': [100000]})}), 'Computer Vision Software Engineer': defaultdict(<function load_sa
laries.<locals>.<lambda> at 0x000002B839AD7A60>, {2021: defaultdict(<function load_s
alaries.<locals>.<lambda> at 0x000002B839AD7B00>, {'US': [70000, 95746]}), 2022: def
aultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839ADB380>, {'AU': [1
50000]})}), 'Director of Data Engineering': defaultdict(<function load_salaries.<loc
als>.<lambda> at 0x000002B839AD7C40>, {2021: defaultdict(<function load_salaries.<lo
cals>.<lambda> at 0x000002B839AD7CE0>, {'GB': [113476], 'US': [200000]})}), 'Data Sc
ience Engineer': defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B8
39AD7EC0>, {2021: defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B
839AD7F60>, {'GR': [40189], 'CA': [127221]}), 2022: defaultdict(<function load_salar
ies.<locals>.<lambda> at 0x000002B839ADA660>, {'MX': [60000]})}), 'Principal Data En
gineer': defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD8180
>, {2021: defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD822
0>, {'US': [200000, 185000, 600000]})}), 'Machine Learning Developer': defaultdict(<
function load_salaries.<locals>.<lambda> at 0x000002B839AD84A0>, {2021: defaultdict
(<function load_salaries.<locals>.<lambda> at 0x000002B839AD8540>, {'IQ': [10000
0]}), 2022: defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839ADA
840>, {'CA': [78791, 78791]})}), 'Applied Machine Learning Scientist': defaultdict(<
function load_salaries.<locals>.<lambda> at 0x000002B839AD8680>, {2021: defaultdict
(<function load_salaries.<locals>.<lambda> at 0x000002B839AD8720>, {'US': [38400, 42
3000]}), 2022: defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839
ADB100>, {'CZ': [31875], 'US': [75000]})}), 'Data Analytics Manager': defaultdict(<f
unction load_salaries.<locals>.<lambda> at 0x000002B839AD8A40>, {2021: defaultdict(<
function load_salaries.<locals>.<lambda> at 0x000002B839AD8AE0>, {'US': [120000, 120
000, 140000]}), 2022: defaultdict(<function load_salaries.<locals>.<lambda> at 0x000
002B839ADA5C0>, {'US': [145000, 105400, 150260, 109280]})}), 'Head of Data Science':
defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD8C20>, {2021:
defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD8CC0>, {'RU':
[85000], 'US': [110000]}), 2022: defaultdict(<function load_salaries.<locals>.<lambd
a> at 0x000002B839AD9C60>, {'US': [224000, 167875]})}), 'Data Specialist': defaultdi
ct(<function load_salaries.<locals>.<lambda> at 0x000002B839AD8E00>, {2021: defaultd
ict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD8EA0>, {'US': [16500
0]})}), 'Data Architect': defaultdict(<function load_salaries.<locals>.<lambda> at 0
x000002B839AD8FE0>, {2021: defaultdict(<function load_salaries.<locals>.<lambda> at
0x000002B839AD9080>, {'US': [150000, 170000, 180000]}), 2022: defaultdict(<function
load_salaries.<locals>.<lambda> at 0x000002B839AD9F80>, {'CA': [192400, 90700], 'U
S': [208775, 147800, 266400, 213120, 192564, 144854]})}), 'Finance Data Analyst': de
faultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD91C0>, {2021: d
efaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD9260>, {'GB':
[61896]})}), 'Principal Data Analyst': defaultdict(<function load_salaries.<locals>.
<lambda> at 0x000002B839AD9440>, {2021: defaultdict(<function load_salaries.<locals
>.<lambda> at 0x000002B839AD94E0>, {'US': [170000]}), 2022: defaultdict(<function lo
ad_salaries.<locals>.<lambda> at 0x000002B839ADB240>, {'CA': [75000]})}), 'Big Data
Architect': defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD9
620>, {2021: defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD
```

```
96C0>, {'CA': [99703]})}), 'Staff Data Scientist': defaultdict(<function load_salari
es.<locals>.<lambda> at 0x000002B839AD98A0>, {2021: defaultdict(<function load_salar
ies.<locals>.<lambda> at 0x000002B839AD9940>, {'US': [105000]})}), 'Analytics Engine
er': defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD9D00>,
{2022: defaultdict(<function load_salaries.<locals>.<lambda> at 0x000002B839AD9DA0>,
{'US': [175000, 135000, 205300, 184700]})}), 'ETL Developer': defaultdict(<function
load_salaries.<locals>.<lambda> at 0x000002B839ADA0C0>, {2022: defaultdict(<function
load_salaries.<locals>.<lambda> at 0x000002B839ADA160>, {'GR': [54957, 54957]})}),
'Head of Machine Learning': defaultdict(<function load_salaries.<locals>.<lambda> at
0x000002B839ADA340>, {2022: defaultdict(<function load_salaries.<locals>.<lambda> at
0x000002B839ADA3E0>, {'IN': [79039]})}), 'NLP Engineer': defaultdict(<function load_
salaries.<locals>.<lambda> at 0x000002B839ADAA20>, {2022: defaultdict(<function load
_salaries.<locals>.<lambda> at 0x000002B839ADAAC0>, {'US': [37236]})}), 'Lead Machin
e Learning Engineer': defaultdict(<function load_salaries.<locals>.<lambda> at 0x000
002B839ADAC00>, {2022: defaultdict(<function load_salaries.<locals>.<lambda> at 0x00
0002B839ADACA0>, {'DE': [87932]})}), 'Data Analytics Lead': defaultdict(<function lo
ad_salaries.<locals>.<lambda> at 0x000002B839ADB4C0>, {2022: defaultdict(<function l
oad_salaries.<locals>.<lambda> at 0x000002B839ADB560>, {'US': [405000]})})})
```

c.3) Un print legible

En la comprobación anterior, puedes observar que yo he utilizado un diccionario por defecto dentro de otro diccionario por defecto. Pero la mezcla de información impide verla con claridad. Seguramente puedes tú mostrarla de manera más legible con unas pocas instrucciones:

In [22]:

```python
# Esta celda debe ser completada por el estudiante

def to_string (dictionary):
    for key_1 in dictionary:
        for key_2 in dictionary[key_1]:
            for key_3 in dictionary[key_1][key_2]:
                print(key_1 + " " + str(key_2) + " " + key_3 + " -> " + str(diction

to_string(Salarios)
```

```
Data Scientist 2020 DE -> [79833, 62726, 49268]
Data Scientist 2020 HU -> [35735]
Data Scientist 2020 FR -> [51321, 39916, 42197]
Data Scientist 2020 IN -> [40481]
Data Scientist 2020 US -> [68428, 45760, 105000, 118000, 120000, 138350, 412000, 105
000]
Data Scientist 2020 GB -> [76958]
Data Scientist 2020 ES -> [38776]
Data Scientist 2020 IT -> [21669]
Data Scientist 2020 AT -> [91237]
Data Scientist 2020 LU -> [62726]
Data Scientist 2021 FR -> [53192, 49646, 36643, 77684]
Data Scientist 2021 IN -> [29751, 9466, 33808, 28399, 16904]
Data Scientist 2021 US -> [73000, 100000, 80000, 82500, 150000, 5679, 147000, 16000
0, 135000, 165000, 115000, 90000, 130000, 100000, 58000, 109000]
Data Scientist 2021 NG -> [50000]
Data Scientist 2021 CA -> [75774, 87738, 103691]
Data Scientist 2021 UA -> [13400]
Data Scientist 2021 IL -> [119059]
Data Scientist 2021 MX -> [2859]
Data Scientist 2021 CL -> [40038]
Data Scientist 2021 DE -> [90734, 90734, 88654, 25532]
Data Scientist 2021 AT -> [61467]
Data Scientist 2021 ES -> [37825, 46809]
Data Scientist 2021 BR -> [12901]
Data Scientist 2021 GB -> [116914, 56256]
Data Scientist 2021 VN -> [4000]
Data Scientist 2021 TR -> [20171]
Data Scientist 2022 US -> [130000, 90000, 136620, 99360, 146000, 123000, 165220, 120
160, 180000, 120000, 95550, 167000, 123000, 150000, 211500, 138600, 170000, 123000,
215300, 158200, 180000, 260000, 180000, 80000, 140400, 215300, 104890, 140000, 22000
0, 140000, 185100, 200000, 120000, 230000, 100000, 100000, 165000, 48000, 135000, 78
000, 141300, 102100, 205300, 140400, 176000, 144000, 205300, 140400, 140000, 210000,
140000, 210000, 140000, 210000, 140000, 230000, 150000, 210000, 160000, 130000]
Data Scientist 2022 GB -> [117789, 104702, 65438, 39263, 71982, 45807, 183228, 9161
4]
Data Scientist 2022 IN -> [31615, 18442]
Data Scientist 2022 DZ -> [100000]
Data Scientist 2022 PL -> [35590]
Data Scientist 2022 CA -> [52396, 69336]
Data Scientist 2022 MY -> [40000]
Data Scientist 2022 AU -> [86703]
Data Scientist 2022 CH -> [122346]
Machine Learning Scientist 2020 JP -> [260000]
Machine Learning Scientist 2021 US -> [225000, 120000]
Machine Learning Scientist 2021 PK -> [12000]
Machine Learning Scientist 2021 CA -> [225000]
Machine Learning Scientist 2022 US -> [160000, 112300, 153000]
Big Data Engineer 2020 GB -> [109024, 114047]
Big Data Engineer 2020 US -> [70000]
Big Data Engineer 2021 RO -> [60000]
Big Data Engineer 2021 IN -> [22611, 16228]
Big Data Engineer 2021 MD -> [18000]
Big Data Engineer 2021 CH -> [5882]
Product Data Analyst 2020 HN -> [20000]
Product Data Analyst 2020 IN -> [6072]
```

```
Machine Learning Engineer 2020 US -> [150000, 250000, 138000]
Machine Learning Engineer 2020 CN -> [43331]
Machine Learning Engineer 2020 HR -> [45618]
Machine Learning Engineer 2021 ES -> [47282]
Machine Learning Engineer 2021 DE -> [94564, 24823, 85000]
Machine Learning Engineer 2021 IN -> [20000, 24342, 66265]
Machine Learning Engineer 2021 BE -> [82744, 88654]
Machine Learning Engineer 2021 US -> [125000, 81000, 200000, 185000]
Machine Learning Engineer 2021 JP -> [74000]
Machine Learning Engineer 2021 CO -> [21844]
Machine Learning Engineer 2021 SI -> [24823]
Machine Learning Engineer 2021 PL -> [46597]
Machine Learning Engineer 2021 IT -> [51064]
Machine Learning Engineer 2022 US -> [189650, 164996, 189650, 164996, 120000, 22000
0, 120000, 214000, 192600]
Machine Learning Engineer 2022 GB -> [37300, 124333, 98158]
Machine Learning Engineer 2022 DE -> [87932]
Machine Learning Engineer 2022 AE -> [120000, 65000]
Machine Learning Engineer 2022 NL -> [62651]
Machine Learning Engineer 2022 AU -> [87425]
Machine Learning Engineer 2022 IE -> [71444]
Data Analyst 2020 US -> [72000, 85000, 91000]
Data Analyst 2020 PK -> [8000]
Data Analyst 2020 FR -> [46759]
Data Analyst 2020 NG -> [10000]
Data Analyst 2020 IN -> [6072]
Data Analyst 2021 US -> [80000, 80000, 75000, 62000, 90000, 50000, 90000, 135000, 60
000, 200000, 80000, 93000]
Data Analyst 2021 FR -> [59102]
Data Analyst 2021 GB -> [51519]
Data Analyst 2021 ES -> [10354]
Data Analyst 2021 CA -> [71786]
Data Analyst 2021 DE -> [63831]
Data Analyst 2022 US -> [155000, 120600, 102100, 84900, 99000, 116000, 106260, 12650
0, 90320, 124190, 130000, 110000, 170000, 115500, 112900, 90320, 112900, 90320, 1670
00, 136600, 109280, 135000, 58000, 132000, 128875, 93700, 164000, 112900, 90320, 115
934, 81666, 58000, 135000, 50000, 135000, 100000, 90320, 112900, 115934, 81666, 9905
0, 116150, 170000, 80000, 100000, 69000, 150075, 126500, 106260, 105000, 110925, 990
00, 60000, 170000, 129000, 150000]
Data Analyst 2022 CA -> [130000, 61300, 130000, 61300, 85000, 75000, 67000, 52000]
Data Analyst 2022 GB -> [52351, 39263, 65438, 45807]
Data Analyst 2022 ES -> [43966, 32974]
Data Analyst 2022 GR -> [43966, 32974, 20000]
Lead Data Scientist 2020 US -> [190000]
Lead Data Scientist 2020 AE -> [115000]
Lead Data Scientist 2021 IN -> [40570]
Business Data Analyst 2020 US -> [135000, 100000]
Business Data Analyst 2021 LU -> [59102]
Business Data Analyst 2022 IN -> [18442]
Business Data Analyst 2022 CA -> [70912]
Lead Data Engineer 2020 NZ -> [125000]
Lead Data Engineer 2020 US -> [56000]
Lead Data Engineer 2021 US -> [276000, 160000]
Lead Data Engineer 2021 GB -> [103160]
Lead Data Engineer 2022 CA -> [118187]
Lead Data Analyst 2020 US -> [87000]
```

```
Lead Data Analyst 2021 US -> [170000]
Lead Data Analyst 2021 IN -> [19609]
Data Engineer 2020 JP -> [41689]
Data Engineer 2020 GR -> [47899]
Data Engineer 2020 MX -> [33511]
Data Engineer 2020 AT -> [74130]
Data Engineer 2020 US -> [106000, 188000, 110000, 130800]
Data Engineer 2020 GB -> [112872]
Data Engineer 2020 FR -> [70139]
Data Engineer 2020 DE -> [54742]
Data Engineer 2021 US -> [140000, 150000, 115000, 150000, 200000, 100000, 90000, 800
00, 26005, 165000, 20000, 110000, 200000, 93150, 111775, 72500, 112000]
Data Engineer 2021 PL -> [28476]
Data Engineer 2021 IN -> [30428, 21637]
Data Engineer 2021 NL -> [45391, 69741]
Data Engineer 2021 MT -> [28369]
Data Engineer 2021 GB -> [82528, 76833, 66022, 72212, 96282]
Data Engineer 2021 IR -> [4000]
Data Engineer 2021 TR -> [12103, 28016]
Data Engineer 2021 DE -> [65013]
Data Engineer 2022 US -> [135000, 170000, 150000, 242000, 200000, 181940, 132320, 22
0110, 160080, 165400, 132320, 243900, 128875, 93700, 156600, 108800, 113000, 160000,
136000, 165400, 136994, 101570, 132320, 155000, 209100, 154600, 175000, 183600, 6390
0, 82900, 100800, 209100, 154600, 180000, 80000, 105000, 120000, 100000, 324000, 216
000, 210000, 115000, 65000, 155000, 206699, 99100, 130000, 115000, 110500, 130000, 1
60000, 200100, 160000, 145000, 70500, 175100, 140250, 54000, 100000, 25000, 220110,
160080, 154000, 126000]
Data Engineer 2022 GB -> [52351, 78526, 52351, 45807, 78526, 65438, 117789, 98158, 7
8526, 58894, 104702, 91614, 98158, 78526]
Data Engineer 2022 GR -> [65949, 49461, 87932, 76940]
Data Engineer 2022 ES -> [49461, 87932, 76940, 65949]
Data Engineer 2022 DE -> [54957, 58035]
Data Engineer 2022 FR -> [68147]
Data Science Consultant 2020 IN -> [5707]
Data Science Consultant 2020 US -> [103000]
Data Science Consultant 2021 DE -> [76833, 63831, 76833]
Data Science Consultant 2021 ES -> [69741]
Data Science Consultant 2021 US -> [90000]
BI Data Analyst 2020 US -> [98000]
BI Data Analyst 2021 US -> [150000, 100000, 36259, 55000]
BI Data Analyst 2021 KE -> [9272]
Director of Data Science 2020 US -> [325000]
Director of Data Science 2021 PL -> [153667]
Director of Data Science 2021 JP -> [168000]
Director of Data Science 2021 DE -> [130026, 141846]
Director of Data Science 2021 US -> [250000]
Director of Data Science 2022 CA -> [196979]
Research Scientist 2020 NL -> [42000]
Research Scientist 2020 US -> [450000]
Research Scientist 2021 GB -> [82528]
Research Scientist 2021 CA -> [187442, 96113, 63810]
Research Scientist 2021 FR -> [62649, 56738]
Research Scientist 2021 PT -> [60757]
Research Scientist 2021 US -> [50000]
Research Scientist 2021 CN -> [100000]
Research Scientist 2021 CZ -> [69999]
```

```
Research Scientist 2022 US -> [144000, 120000]
Research Scientist 2022 FR -> [93427]
Research Scientist 2022 AT -> [64849]
Machine Learning Manager 2020 CA -> [117104]
Data Engineering Manager 2020 DE -> [59303]
Data Engineering Manager 2020 ES -> [79833]
Data Engineering Manager 2021 US -> [150000, 153000, 174000]
Machine Learning Infrastructure Engineer 2020 PT -> [50180]
Machine Learning Infrastructure Engineer 2021 US -> [195000]
Machine Learning Infrastructure Engineer 2022 PT -> [58255]
ML Engineer 2020 DE -> [15966]
ML Engineer 2021 US -> [270000, 256000]
ML Engineer 2021 JP -> [63711, 77364]
ML Engineer 2022 PT -> [21983]
AI Scientist 2020 DK -> [45896]
AI Scientist 2021 US -> [12000, 12000]
AI Scientist 2021 AS -> [18053]
AI Scientist 2021 ES -> [55000]
AI Scientist 2022 US -> [120000, 200000]
Computer Vision Engineer 2020 US -> [60000]
Computer Vision Engineer 2021 BR -> [24000, 18907]
Computer Vision Engineer 2021 DK -> [28609]
Computer Vision Engineer 2022 US -> [125000]
Computer Vision Engineer 2022 LU -> [10000]
Principal Data Scientist 2020 DE -> [148261]
Principal Data Scientist 2021 US -> [151000, 220000, 235000, 416000]
Principal Data Scientist 2021 DE -> [173762]
Principal Data Scientist 2022 DE -> [162674]
Data Science Manager 2020 US -> [190200]
Data Science Manager 2021 US -> [240000, 144000, 174000, 54094]
Data Science Manager 2021 FR -> [152000]
Data Science Manager 2021 IN -> [94665]
Data Science Manager 2022 US -> [161342, 137141, 241000, 159000, 152500]
Head of Data 2021 US -> [235000]
Head of Data 2021 RU -> [230000]
Head of Data 2021 SI -> [102839]
Head of Data 2022 US -> [200000]
Head of Data 2022 EE -> [32974]
3D Computer Vision Researcher 2021 IN -> [5409]
Data Analytics Engineer 2021 DE -> [79197]
Data Analytics Engineer 2021 US -> [110000]
Data Analytics Engineer 2021 GB -> [50000]
Data Analytics Engineer 2022 PK -> [20000]
Applied Data Scientist 2021 CA -> [54238]
Applied Data Scientist 2021 GB -> [110037]
Applied Data Scientist 2022 US -> [157000, 380000, 177000]
Marketing Data Analyst 2021 DK -> [88654]
Cloud Data Engineer 2021 SG -> [89294]
Cloud Data Engineer 2021 US -> [160000]
Financial Data Analyst 2021 US -> [450000]
Financial Data Analyst 2022 US -> [100000]
Computer Vision Software Engineer 2021 US -> [70000, 95746]
Computer Vision Software Engineer 2022 AU -> [150000]
Director of Data Engineering 2021 GB -> [113476]
Director of Data Engineering 2021 US -> [200000]
Data Science Engineer 2021 GR -> [40189]
```

```
Data Science Engineer 2021 CA -> [127221]
Data Science Engineer 2022 MX -> [60000]
Principal Data Engineer 2021 US -> [200000, 185000, 600000]
Machine Learning Developer 2021 IQ -> [100000]
Machine Learning Developer 2022 CA -> [78791, 78791]
Applied Machine Learning Scientist 2021 US -> [38400, 423000]
Applied Machine Learning Scientist 2022 CZ -> [31875]
Applied Machine Learning Scientist 2022 US -> [75000]
Data Analytics Manager 2021 US -> [120000, 120000, 140000]
Data Analytics Manager 2022 US -> [145000, 105400, 150260, 109280]
Head of Data Science 2021 RU -> [85000]
Head of Data Science 2021 US -> [110000]
Head of Data Science 2022 US -> [224000, 167875]
Data Specialist 2021 US -> [165000]
Data Architect 2021 US -> [150000, 170000, 180000]
Data Architect 2022 CA -> [192400, 90700]
Data Architect 2022 US -> [208775, 147800, 266400, 213120, 192564, 144854]
Finance Data Analyst 2021 GB -> [61896]
Principal Data Analyst 2021 US -> [170000]
Principal Data Analyst 2022 CA -> [75000]
Big Data Architect 2021 CA -> [99703]
Staff Data Scientist 2021 US -> [105000]
Analytics Engineer 2022 US -> [175000, 135000, 205300, 184700]
ETL Developer 2022 GR -> [54957, 54957]
Head of Machine Learning 2022 IN -> [79039]
NLP Engineer 2022 US -> [37236]
Lead Machine Learning Engineer 2022 DE -> [87932]
Data Analytics Lead 2022 US -> [405000]
```

c.4) Sueldo medio por grupos de puesto, nivel y año

Define ahora una función `sueldo_medio_agrupando` que, partiendo de la *tabla compacta* generada, proporcione el sueldo medio de un puesto de trabajo para un nivel y año dado.

In [23]:
```python
# Esta celda debe ser completada por el estudiante
def sueldo_medio_agrupando(compact_table: Dict[tuple, List[int]], employment: str,
    """
    Calculate the mean salary for a specific job title, year, and difficulty level

    This function iterates through a given dictionary of job data, filters entries
    year, and difficulty level, and calculates the mean salary for the matching ent

    Parameters
    ----------
    compact_table : dict
        A dictionary containing job data in the form of key-value pairs, where the
        structure (job_title, job_difficulty, job_year, job_country), and the value

    employment : str
        A substring that is used to filter job titles. Job titles starting with thi

    level : int
        The difficulty level for which salaries should be considered.
```

```
    year : int
        The specific year for which salaries should be considered.

    Returns
    -------
    float or None
        The mean salary for the specified job title, year, and difficulty level, or
    """
    matching_salaries = []

    for key in compact_table:
        job_title, job_difficulty, job_year, job_country = key
        if (
            employment in job_title and
            job_year == year and
            job_difficulty == level
        ):
            matching_salaries.extend(compact_table[key])

    if matching_salaries:
        mean_salary = sum(matching_salaries) / len(matching_salaries)
        return round(mean_salary, 1)
    else:
        return None  # Return None if no matching data is found
```

In [24]:
```
# Comprobación:

for cargo in ["Data Sci", "Machine", "Data Engi"]:
    print(cargo, sueldo_medio_agrupando(Salarios_tabla_compact, cargo, 3, 2022))
```

```
Data Sci 161890.3
Machine 138693.6
Data Engi 140939.5
```

c.5) Un cálculo con la tabla anterior

Cálculo del sueldo medio de un puesto de trabajo en un año y país dados. Para facilitar la lectura, redondeamos a dos decimales las medias. En esta función, debes tener cuidado con las situaciones posibles en que no existen salarios, pues la media se calcularçia erróneamente.

In [25]:
```
# Esta celda debe ser completada por el estudiante
def average_salary_with_dict(table_of_tables: Dict[str, Dict[int, Dict[str, List[in
    """
    Calculate the average salary for a specific job, year, and country from a neste

    Parameters:
    - table_of_tables (Dict[str, Dict[int, Dict[str, List[int]]]]): A nested diction
    - job (str): The job title.
    - year (int): The year for which you want to calculate the average salary.
    - country (str): The country for which you want to calculate the average salary

    Returns:
```

```
    - Optional[float]: The average salary for the specified job, year, and country,
      Returns 0 if the list of salaries is empty or there is a missing field in the
    """
    try:
        matching_salaries = table_of_tables[job][year][country]
    except:
        return 0

    if not matching_salaries:
        return 0

    mean_salary = sum(matching_salaries) / len(matching_salaries)
    return round(mean_salary, 2)
```

In [26]:
```
# Comprobación de funcionamiento:

for anno in range(2020, 2024):
    print(anno, average_salary_with_dict(Salarios, "Data Scientist", anno, "US"))
```

```
2020 139067.25
2021 106261.19
2022 153483.33
2023 0
```

Nota. Observa que, si la tabla no contiene la información para un año (ej, 2023), la función da un cero, y no un error.

En este apartado se ha profundizado en el conocimiento de diccionarios en Python y se ha logrado completar con éxito cada sub apartado.

d) Algunas gráficas [1 punto]

d.1) Un modelo típico de gráfica

Vamos a diseñar un modelo de gráfica sencillo que nos sirva para las siguientes representaciones. Tomará como parámetro una lista de pares (x, y), y opcionalmente los tres rótulos explicativos que necesitamos incluir. Además, queremos que las etiquetas de las abcisas aparezcan inclinadas, para poder luego mostrar intervalos de edad.

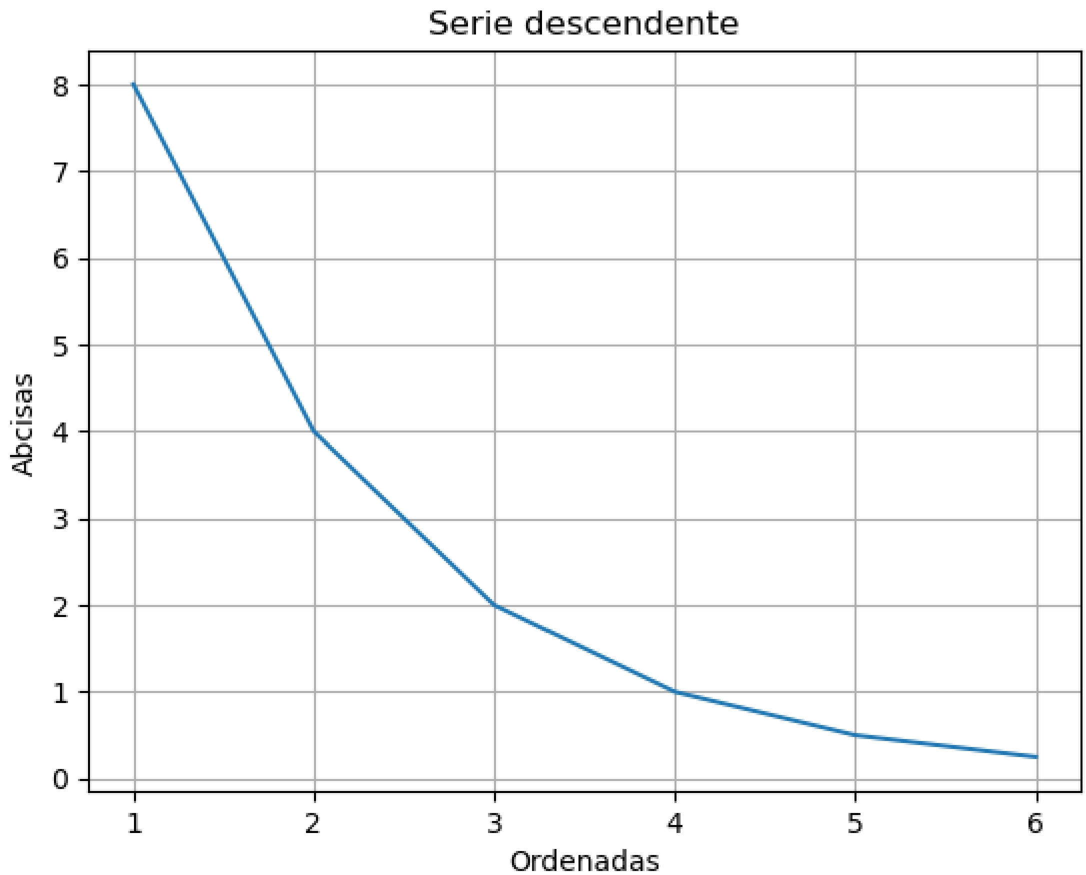
Las pruebas de funcionamiento te darán más información que las explicaciones que pueda yo dar aquí.

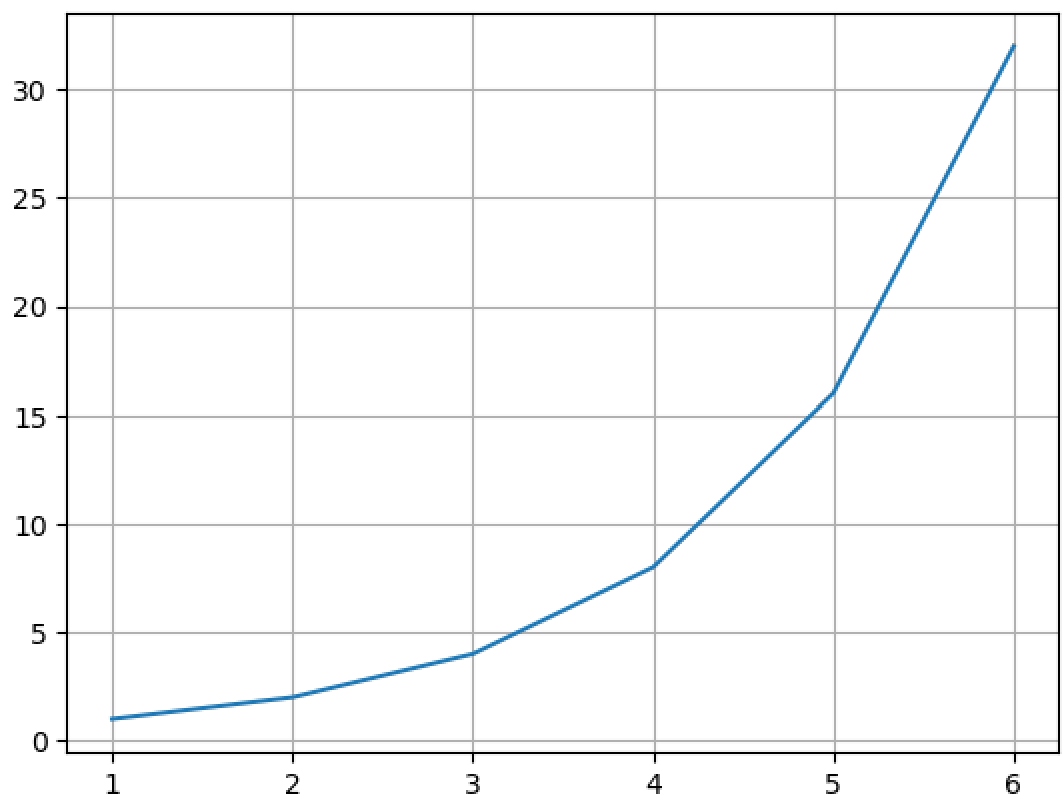
In [27]:
```
# Esta celda debe ser completada por el estudiante

def representar_xxx_yyy(data, title_labels=None):
    """
    Plot data points and add legends to a Matplotlib plot.

    Parameters:
    - data (list of tuples): List of (x, y) data points to be plotted.
```

```
    - legend_labels (list of str): List of legend labels corresponding to the data

    Example:
    representar_xxx_yyy([(1, 8), (2, 4), (3, 2), (4, 1), (5, 0.5), (6, 0.25)],
                        ["Serie descendente", "Ordenadas", "Abcisas"])
    """
    # Unzip the data points into separate lists for x and y coordinates
    x_values, y_values = zip(*data)

    # Create a Matplotlib figure and axis
    fig, ax = plt.subplots()

    # Plot the data points
    ax.plot(x_values, y_values)

    if title_labels:
        # Set title and axis labels
        ax.set_title(title_labels[0])
        ax.set_xlabel(title_labels[1])
        ax.set_ylabel(title_labels[2])

    # Add grid
    ax.grid(True)
    ax.xaxis.set_major_locator(MaxNLocator(integer=True))

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

In [28]:

```
# Pruebas de funcionamiento:

representar_xxx_yyy([(1, 8), (2, 4), (3, 2), (4, 1), (5, 0.5), (6, 0.25)], ["Serie

representar_xxx_yyy([(1, 1), (2, 2), (3, 4), (4, 8), (5, 16), (6, 32)])
```

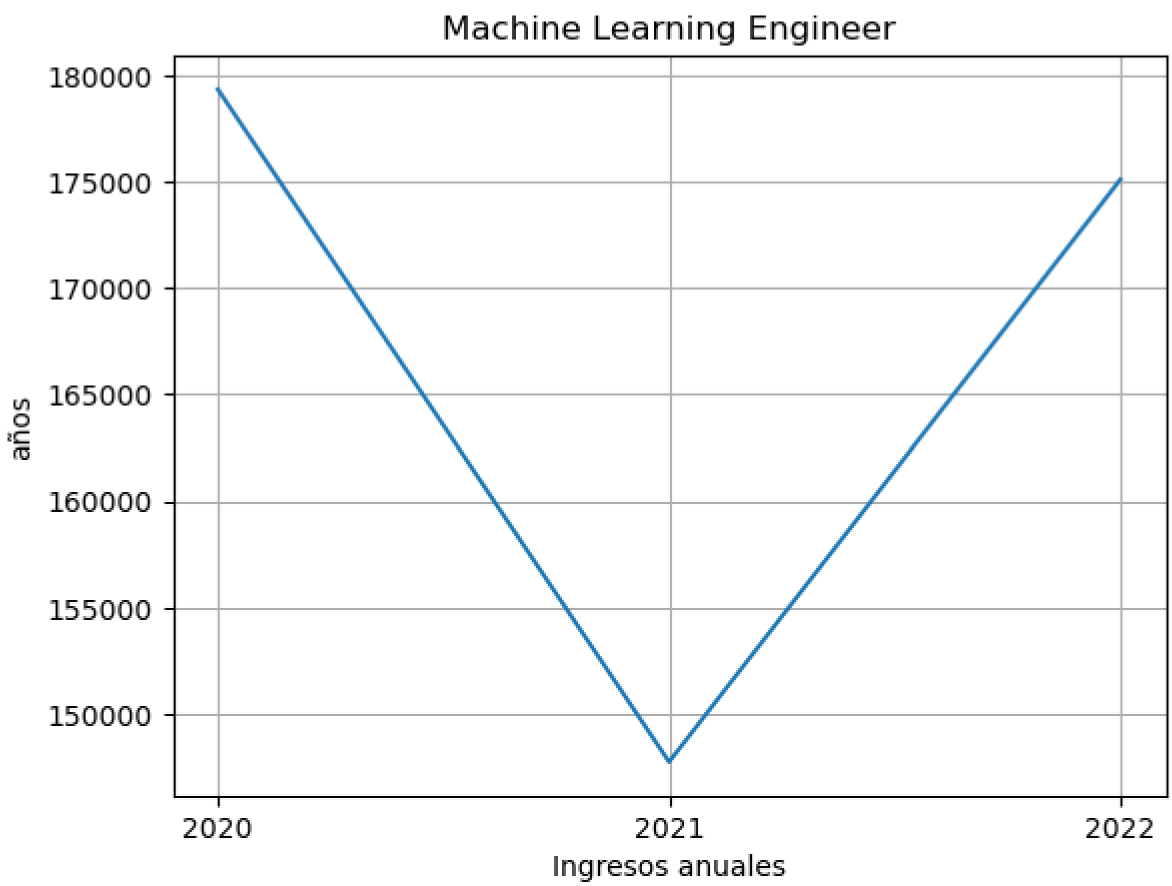

Serie descendente
Abcisas
Ordenadas
0
1
2
3
4
5
6
7
8
1
2
3
4
5
6


0
5
10
15
20
25
30
1
2
3
4
5
6

Lógicamente, hemos diseñado nuestro modelo para aplicarlo posteriormente a los datos que ya tenemos. Concretamente, podemos aplicarlo también a la representación de los sueldos medios registrados en cada año.

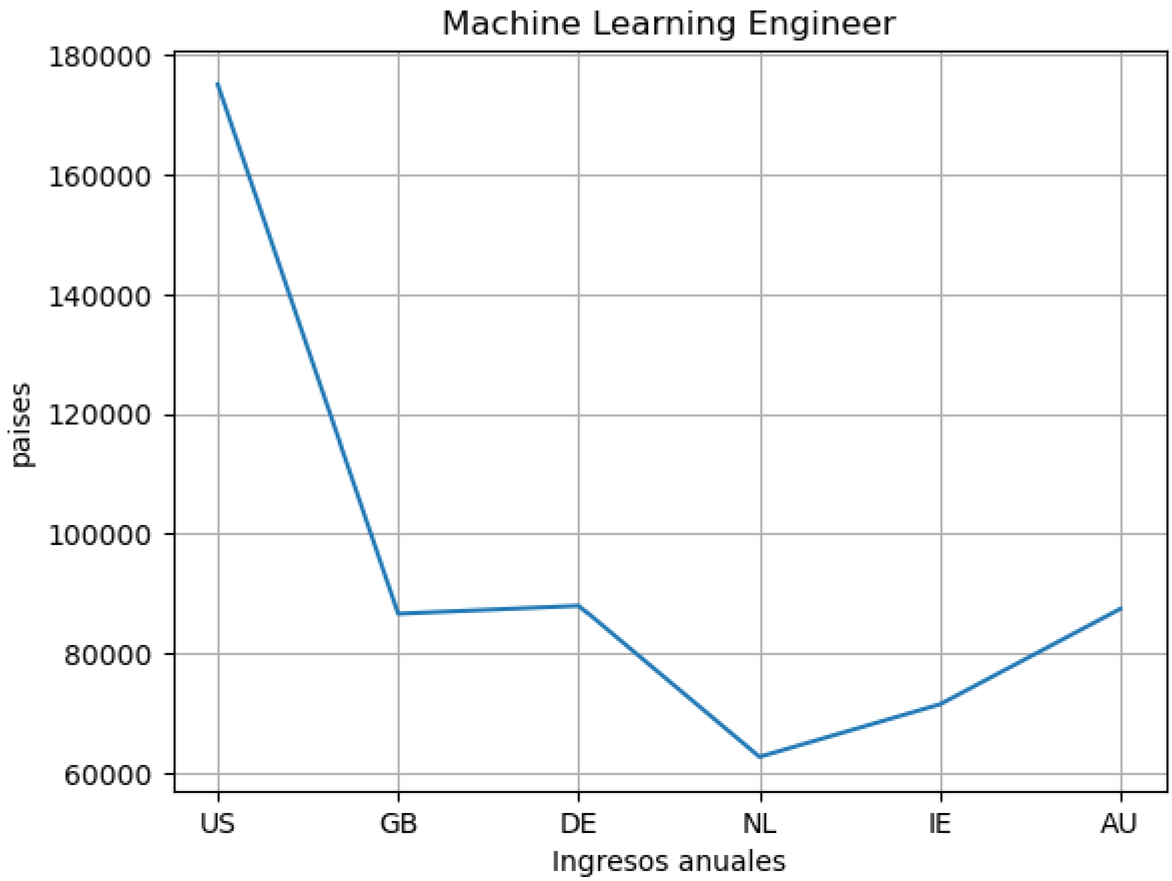
In [29]:
```
# Pruebas de funcionamiento:

annos = range(2020, 2023)
annos_sueldos = [(anno, average_salary_with_dict(Salarios, "Machine Learning Engine

print(annos_sueldos)

representar_xxx_yyy(annos_sueldos, ["Machine Learning Engineer", "Ingresos anuales"
```

```
[(2020, 179333.33), (2021, 147750.0), (2022, 175099.11)]
```



In [30]:
```
# Pruebas de funcionamiento:

paises = ["US", "GB", "DE", "NL", "IE", "AU"]

paises_sueldos = [(pais, average_salary_with_dict(Salarios, "Machine Learning Engin

print(annos_sueldos)

representar_xxx_yyy(paises_sueldos, ["Machine Learning Engineer", "Ingresos anuales
```

```
[(2020, 179333.33), (2021, 147750.0), (2022, 175099.11)]
```

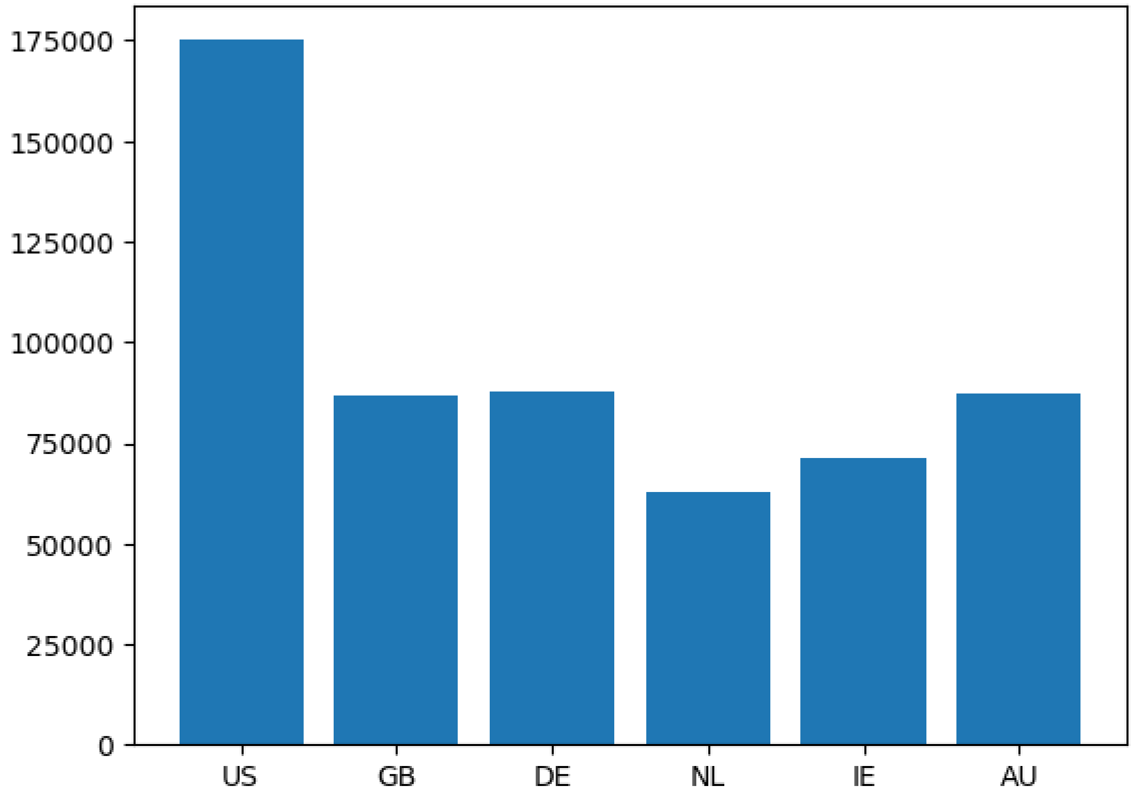


d.2) Histograma

Un gráfico más adecuado para este cometido es el histograma.

Las pruebas de funcionamiento te darán más información que las explicaciones que pueda yo dar aquí.

In [31]:
```
# Esta celda debe ser completada por el estudiante

# Extract the string labels and integer values
labels, values = zip(*paises_sueldos)

# Create a range of numerical values for the x-axis
x = np.arange(len(labels))

# Create a bar plot using the numerical values
plt.bar(x, values)

# Set the x-tick labels as the original string labels
plt.xticks(x, labels)

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



Nota. Vemos que la curva se comporta de un modo extraño, pues sufre una caída en 2001: esto es lo que indican efectivamente los datos.

En este apartado se ha profundizado en el conocimiento de matplotlib y se ha logrado completar con éxito todos los ejercicios. Como único detalle pendiente, los tamaños de los gráficos parecen no ser iguales que en el notebook original, salvo eso los gráficos resultantes son idénticos.

e) Operaciones con dataframes [2 puntos]

En este apartado, vamos a trabajar con tablas de la librería `pandas`, llamadas `dataframes`.

e.1) Carga del dataframe

La primera operación que necesitamos es cargar el archivo de datos en una tabla, como se ve en el siguiente ejemplo.

In [32]:
```
# Esta celda debe ser completada por el estudiante
def load_dataframe(csv_file):
    """
    Load data from a CSV file into a Pandas DataFrame.

    Parameters:
```

```
    - csv_file (str): Path to the CSV file.

    Returns:
    - df (pd.DataFrame): The loaded DataFrame.
    """
    try:
        df = pd.read_csv(csv_file, delimiter= ';')
        return df
    except Exception as e:
        print("Error loading data:", str(e))
        return None
```

In [33]:
```
# Comprobación

tabla_completa = load_dataframe(DatosPunComas)
tabla_completa
```

Out[33]:

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

607 rows × 12 columns

e.2) Ajustes en nuestro archivo de datos

Deseamos ahora prescidir de la primera columna, pues únicamente da un número de orden de las filas, así como de las columnas relativas a la moneda local (`salary` y `salary currency`).

In [34]:
```
# Esta celda debe ser completada por el estudiante
# Delete first column
tabla_abreviada = tabla_completa.drop(tabla_completa.columns[0], axis=1)
columns_to_drop = ['salary', 'salary_currency']
# Drop salary and salary_currency columns
tabla_abreviada = tabla_abreviada.drop(columns=columns_to_drop)
```

In [35]:
```
# Comprobación

tabla_abreviada
```

Out[35]:

	work_year	experience_level	employment_type	job_title	salary_in_usd	employee_resi
0	2020	MI	FT	Data Scientist	79833	
1	2020	SE	FT	Machine Learning Scientist	260000	
2	2020	SE	FT	Big Data Engineer	109024	
3	2020	MI	FT	Product Data Analyst	20000	
4	2020	SE	FT	Machine Learning Engineer	150000	
...	...	...	...	...	...	
602	2022	SE	FT	Data Engineer	154000	
603	2022	SE	FT	Data Engineer	126000	
604	2022	SE	FT	Data Analyst	129000	
605	2022	SE	FT	Data Analyst	150000	
606	2022	MI	FT	AI Scientist	200000	

607 rows × 9 columns

Comprobamos también los tipos de datos de las columnas, para asegurarnos de que los datos numéricos se han cargado como tales; de lo contrario, deberíamos cambiar su tipo.

In [36]:

```
# Comprobación

tabla_abreviada.dtypes
```

Out[36]:

```
work_year              int64
experience_level      object
employment_type       object
job_title             object
salary_in_usd          int64
employee_residence    object
remote_ratio           int64
company_location      object
company_size          object
dtype: object
```

Aunque sólo sea a efectos didácticos, la columna de los porcentajes debería ser un real... Cambia esto, sólo para practicar.

In [37]:
```
# Esta celda debe ser completada por el estudiante
tabla_abreviada['remote_ratio'] = tabla_abreviada['remote_ratio'].astype(float)
```

In [38]:
```
# Comprobación

tabla_abreviada.dtypes
```

Out[38]:
```
work_year                int64
experience_level        object
employment_type         object
job_title               object
salary_in_usd            int64
employee_residence      object
remote_ratio           float64
company_location        object
company_size            object
dtype: object
```

In [39]:
```
# Comprobación

tabla_abreviada
```

Out[39]:

	work_year	experience_level	employment_type	job_title	salary_in_usd	employee_resi
0	2020	MI	FT	Data Scientist	79833	
1	2020	SE	FT	Machine Learning Scientist	260000	
2	2020	SE	FT	Big Data Engineer	109024	
3	2020	MI	FT	Product Data Analyst	20000	
4	2020	SE	FT	Machine Learning Engineer	150000	
...	...	...	...	...	...	
602	2022	SE	FT	Data Engineer	154000	
603	2022	SE	FT	Data Engineer	126000	
604	2022	SE	FT	Data Analyst	129000	
605	2022	SE	FT	Data Analyst	150000	
606	2022	MI	FT	AI Scientist	200000	

607 rows × 9 columns

También, la columna de la experiencia será más manejable si convertimos los código en números (así: "EN" -> 0, "MI" -> 1, "EX" -> 2, "SE" -> 3) y algo parecido haremos con el tamaño de las compañías ("S" -> 1, "EX: 0, "M" -> 2, "L" -> 2).

In [40]:

```
# Esta celda debe ser completada por el estudiante
experience_level_dict = { "EN": 0, "MI": 1, "EX": 2, "SE": 3}
company_size_dict = {"S": 1, "EX": 0, "M": 2, "L": 3}

tabla_abreviada['experience_level'] = tabla_abreviada['experience_level'].map(exper
tabla_abreviada['company_size'] = tabla_abreviada['company_size'].map(company_size_
```

In [41]:

```
# Comprobación

tabla_abreviada
```

Out[41]:

	work_year	experience_level	employment_type	job_title	salary_in_usd	employee_resi
0	2020	1	FT	Data Scientist	79833	
1	2020	3	FT	Machine Learning Scientist	260000	
2	2020	3	FT	Big Data Engineer	109024	
3	2020	1	FT	Product Data Analyst	20000	
4	2020	3	FT	Machine Learning Engineer	150000	
...	...	...	...	...	...	
602	2022	3	FT	Data Engineer	154000	
603	2022	3	FT	Data Engineer	126000	
604	2022	3	FT	Data Analyst	129000	
605	2022	3	FT	Data Analyst	150000	
606	2022	1	FT	AI Scientist	200000	

607 rows × 9 columns

In [42]:

```
# Esta celda debe ser completada por el estudiante
def average_salary_with_dataframe(df: pd.DataFrame, job: str, year: int, country: s
    """
    Calculate the average salary for a specific job, year, and country from a panda

    Parameters:
    - df (pd.DataFrame): A DataFrame containing salary information.
    - job (str): The job title.
    - year (int): The year for which you want to calculate the average salary.
    - country (str): The country for which you want to calculate the average salary

    Returns:
    - Optional[float]: The average salary for the specified job, year, and country,
      Returns 0 if the list of salaries is empty or there is no matching data in th
    """
    try:
        matching_salaries = df[(df['job_title'] == job) & (df['work_year'] == year)
    except (IndexError, KeyError):
```

```
        return 0

    if len(matching_salaries) == 0:
        return 0

    mean_salary = sum(matching_salaries) / len(matching_salaries)
    return round(mean_salary, 2)
```

In [43]:
```
print(average_salary_with_dataframe(tabla_abreviada, "Data Scientist", 2020, "US"))
```

```
139067.25
```

Comprobamos que el resultado es el mismo que el que definimos usando el diccionario:

In [44]:
```
print(average_salary_with_dict(Salarios, "Data Scientist", 2020, "US"))
```

```
139067.25
```

En este apartado se ha profundizado en el conocimiento de la librería de `pandas`. Se han completado con éxito todos los apartados.

f) Un cálculo masivo con map-reduce [Resuelto]

En este apartado se ha de realizar un programa aparte que calcule, para cada país, el número de cada puesto de trabajo que tiene contratado, junto con el máximo sueldo de cada categoría pra dicho país con independencia de laño.

```
C:\...> python puestos_trabajo.py -q ds_salaries.norm.csv
```

El programa funcionará necesariamente con la técnica map-reduce, que podemos poner en juego con la librería `mrjob`.

El funcionamiento del mismo se puede activar también desde aquí:

In [45]:
```
# Hagamos una llamada al programa de consola desde aquí:

! python puestos_trabajo.py -q ds_salaries.norm.csv
```

```
["3D Computer Vision Researcher","IN"]	5409
["AI Scientist","AS"]	18053
["AI Scientist","DK"]	45896
["AI Scientist","ES"]	55000
["AI Scientist","US"]	200000
["Analytics Engineer","US"]	205300
["Applied Data Scientist","CA"]	54238
["Applied Data Scientist","GB"]	110037
["Applied Data Scientist","US"]	380000
["Applied Machine Learning Scientist","CZ"]	31875
["Applied Machine Learning Scientist","US"]	423000
["BI Data Analyst","KE"]	9272
["BI Data Analyst","US"]	150000
["Big Data Architect","CA"]	99703
["Big Data Engineer","CH"]	5882
["Big Data Engineer","GB"]	114047
["Big Data Engineer","IN"]	22611
["Big Data Engineer","MD"]	18000
["Big Data Engineer","RO"]	60000
["Big Data Engineer","US"]	70000
["Business Data Analyst","CA"]	70912
["Business Data Analyst","IN"]	18442
["Business Data Analyst","LU"]	59102
["Business Data Analyst","US"]	135000
["Cloud Data Engineer","SG"]	89294
["Cloud Data Engineer","US"]	160000
["Computer Vision Engineer","BR"]	24000
["Computer Vision Engineer","DK"]	28609
["Computer Vision Engineer","LU"]	10000
["Computer Vision Engineer","US"]	125000
["Computer Vision Software Engineer","AU"]	150000
["Computer Vision Software Engineer","US"]	95746
["Data Analyst","CA"]	130000
["Data Analyst","DE"]	63831
["Data Analyst","ES"]	43966
["Data Analyst","FR"]	59102
["Data Analyst","GB"]	65438
["Data Analyst","GR"]	43966
["Data Analyst","IN"]	6072
["Data Analyst","NG"]	10000
["Data Analyst","PK"]	8000
["Data Analyst","US"]	200000
["Data Analytics Engineer","DE"]	79197
["Data Analytics Engineer","GB"]	50000
["Data Analytics Engineer","PK"]	20000
["Data Analytics Engineer","US"]	110000
["Data Analytics Lead","US"]	405000
["Data Analytics Manager","US"]	150260
["Data Architect","CA"]	192400
["Data Architect","US"]	266400
["Data Engineer","AT"]	74130
["Data Engineer","DE"]	65013
["Data Engineer","ES"]	87932
["Data Engineer","FR"]	70139
["Data Engineer","GB"]	117789
["Data Engineer","GR"]	87932
```

```
["Data Engineer","IN"]  30428
["Data Engineer","IR"]  4000
["Data Engineer","JP"]  41689
["Data Engineer","MT"]  28369
["Data Engineer","MX"]  33511
["Data Engineer","NL"]  69741
["Data Engineer","PL"]  28476
["Data Engineer","TR"]  28016
["Data Engineer","US"]  324000
["Data Engineering Manager","DE"]       59303
["Data Engineering Manager","ES"]       79833
["Data Engineering Manager","US"]       174000
["Data Science Consultant","DE"]        76833
["Data Science Consultant","ES"]        69741
["Data Science Consultant","IN"]        5707
["Data Science Consultant","US"]        103000
["Data Science Engineer","CA"]  127221
["Data Science Engineer","GR"]  40189
["Data Science Engineer","MX"]  60000
["Data Science Manager","FR"]   152000
["Data Science Manager","IN"]   94665
["Data Science Manager","US"]   241000
["Data Scientist","AT"] 91237
["Data Scientist","AU"] 86703
["Data Scientist","BR"] 12901
["Data Scientist","CA"] 103691
["Data Scientist","CH"] 122346
["Data Scientist","CL"] 40038
["Data Scientist","DE"] 90734
["Data Scientist","DZ"] 100000
["Data Scientist","ES"] 46809
["Data Scientist","FR"] 77684
["Data Scientist","GB"] 183228
["Data Scientist","HU"] 35735
["Data Scientist","IL"] 119059
["Data Scientist","IN"] 40481
["Data Scientist","IT"] 21669
["Data Scientist","LU"] 62726
["Data Scientist","MX"] 2859
["Data Scientist","MY"] 40000
["Data Scientist","NG"] 50000
["Data Scientist","PL"] 35590
["Data Scientist","TR"] 20171
["Data Scientist","UA"] 13400
["Data Scientist","US"] 412000
["Data Scientist","VN"] 4000
["Data Specialist","US"]        165000
["Director of Data Engineering","GB"]   113476
["Director of Data Engineering","US"]   200000
["Director of Data Science","CA"]       196979
["Director of Data Science","DE"]       141846
["Director of Data Science","JP"]       168000
["Director of Data Science","PL"]       153667
["Director of Data Science","US"]       325000
["ETL Developer","GR"]  54957
["Finance Data Analyst","GB"]   61896
```

["Financial Data Analyst","US"]	450000
["Head of Data Science","RU"]	85000
["Head of Data Science","US"]	224000
["Head of Data","EE"]	32974
["Head of Data","RU"]	230000
["Head of Data","SI"]	102839
["Head of Data","US"]	235000
["Head of Machine Learning","IN"]	79039
["Lead Data Analyst","IN"]	19609
["Lead Data Analyst","US"]	170000
["Lead Data Engineer","CA"]	118187
["Lead Data Engineer","GB"]	103160
["Lead Data Engineer","NZ"]	125000
["Lead Data Engineer","US"]	276000
["Lead Data Scientist","AE"]	115000
["Lead Data Scientist","IN"]	40570
["Lead Data Scientist","US"]	190000
["Lead Machine Learning Engineer","DE"]	87932
["ML Engineer","DE"]	15966
["ML Engineer","JP"]	77364
["ML Engineer","PT"]	21983
["ML Engineer","US"]	270000
["Machine Learning Developer","CA"]	78791
["Machine Learning Developer","IQ"]	100000
["Machine Learning Engineer","AE"]	120000
["Machine Learning Engineer","AU"]	87425
["Machine Learning Engineer","BE"]	88654
["Machine Learning Engineer","CN"]	43331
["Machine Learning Engineer","CO"]	21844
["Machine Learning Engineer","DE"]	94564
["Machine Learning Engineer","ES"]	47282
["Machine Learning Engineer","GB"]	124333
["Machine Learning Engineer","HR"]	45618
["Machine Learning Engineer","IE"]	71444
["Machine Learning Engineer","IN"]	66265
["Machine Learning Engineer","IT"]	51064
["Machine Learning Engineer","JP"]	74000
["Machine Learning Engineer","NL"]	62651
["Machine Learning Engineer","PL"]	46597
["Machine Learning Engineer","SI"]	24823
["Machine Learning Engineer","US"]	250000
["Machine Learning Infrastructure Engineer","PT"]	58255
["Machine Learning Infrastructure Engineer","US"]	195000
["Machine Learning Manager","CA"]	117104
["Machine Learning Scientist","CA"]	225000
["Machine Learning Scientist","JP"]	260000
["Machine Learning Scientist","PK"]	12000
["Machine Learning Scientist","US"]	225000
["Marketing Data Analyst","DK"]	88654
["NLP Engineer","US"]	37236
["Principal Data Analyst","CA"]	75000
["Principal Data Analyst","US"]	170000
["Principal Data Engineer","US"]	600000
["Principal Data Scientist","DE"]	173762
["Principal Data Scientist","US"]	416000
["Product Data Analyst","HN"]	20000

```
["Product Data Analyst","IN"]   6072
["Research Scientist","AT"]     64849
["Research Scientist","CA"]     187442
["Research Scientist","CN"]     100000
["Research Scientist","CZ"]     69999
["Research Scientist","FR"]     93427
["Research Scientist","GB"]     82528
["Research Scientist","NL"]     42000
["Research Scientist","PT"]     60757
["Research Scientist","US"]     450000
["Staff Data Scientist","US"]   105000
```

In [46]:
```
# Para que el resultado se almacene en un archivo:

! python puestos_trabajo.py -q ds_salaries.norm.csv  > sueldos_maximos.txt
```

Para que pueda yo ver tu programa cómodamente desde aquí, también se puede mostrar con un comando de la consola, anteponiendo el símbolo `!`. Observaciones:

- La instrucción siguiente está comentada para ocultar una solución mía. Tú debes suprimir el símbolo `#` del comentario para mostrar tu solución aquí.
- Desde mac o linux, se ha de usar el comando `cat`, en vez de `type`.

In [47]:
```
! type puestos_trabajo.py
```

```
# -*- coding: utf-8 -*-
"""
@author: CPAREJA
"""

from mrjob.job import MRJob

class MRTotalesPuestos_Salarios(MRJob):

    def mapper(self, _, line):
        linea = line.rstrip().split(";")
        cargo, sueldo, pais = linea[3], linea[4], linea[7]
        yield (cargo, pais), int(sueldo)

    def reducer(self, key, values):
        yield key, max(values)

if __name__ == '__main__':
    MRTotalesPuestos_Salarios.run()
```

f_bis) Un cálculo masivo con map-reduce [0.5 puntos]

Como la solución del apartado anterior se entregó por error mío, puedes, si lo deseas, inventar tú mismo un nuevo enunciado para ser resuelto con la técnica de *map-reduce*, y proponer luego una solución para el mismo.

Enunciado propuesto

Implementa un script de python para que, empleando la técnica map-reduce, obtengamos para un país de residencia de empleado, un año dado y un puesto de trabajo especificado, el número de empleados que hay en los distintos niveles de experiencia (es decir, cuantos empleados hay con nivel de experiencia 1, 2, etc...).

Aclaraciones

Se tomarán todas las líneas que contengan en el puesto de trabajo el puesto especificado por el usuario, es decir, si se introduce como puesto de trabajo *Data Scientist* serán válidos todos los puestos de trabajo que lo incluyan, como por ejemplo *Leader Data Scientist*.

In [48]:
```
# El script implementado es el siguiente:
! type paises_puestos_experiencia.py
```

```
# -*- coding: utf-8 -*-
"""
@author: Alejandro Borrego Megías
"""

from mrjob.job import MRJob

class MRTotalesPuestosSalarios(MRJob):
    def configure_args(self):
        super(MRTotalesPuestosSalarios, self).configure_args()
        self.add_passthru_arg('--country', default='US', help="Indica el código del
país")
        self.add_passthru_arg('--job', default='Data Scientist', help="Indica el pue
sto de trabajo")
        self.add_passthru_arg('--year', default=2020, help="Indica el año")

    def mapper(self, _, line):
        linea = line.rstrip().split(";")
        pais, cargo, anno, experiencia = linea[7], linea[3], linea[0], linea[1]
        if pais == self.options.country and self.options.job in cargo and anno == st
r(self.options.year):
            yield (pais, cargo, anno, experiencia), 1

    def reducer(self, key, values):
        yield key, sum(values)

if __name__ == '__main__':
    MRTotalesPuestosSalarios.run()
```

In [49]:
```
# Ejemplo de ejecución con los parámetros por defecto ( país US, año 2020 y puesto

! python paises_puestos_experiencia.py -q ds_salaries.norm.csv
```

```
["US","Data Scientist","2020","0"]      1
["US","Data Scientist","2020","1"]      4
["US","Data Scientist","2020","3"]      3
["US","Lead Data Scientist","2020","3"] 1
```

In [50]:
```
# Ejemplo de ejecución con país FR, año 2021 y puestos que contengan Scientist

! python paises_puestos_experiencia.py -q ds_salaries.norm.csv --country="FR" --job
```

```
["FR","Data Scientist","2021","0"]      2
["FR","Data Scientist","2021","3"]      2
["FR","Research Scientist","2021","1"]  2
```

In [51]:
```
# Ejemplo de ejecución con país GB, año 2022, y puestos que contengan Machine Learn
! python paises_puestos_experiencia.py -q ds_salaries.norm.csv --country="GB" --job
```

En este apartado se ha experimentado por primera vez con la útil técnica Map-reduce completando con éxito todos los ejercicios propuestos.

g) Un apartado libre [0.5 puntos]

Dejo este apartado a tu voluntad. Inventa tú mismo el enunciado y resuélvelo, mostrando algún aspecto de programación en Python no contemplado o alguna técnica o librería que no has puesto en juego en los apartados anteriores, relacionado con el análisis de datos y con este proyecto. He aquí dos o tres ejemplos posibles:

- Me he quedado un poco insatisfecho con el uso de pandas, que encuentro un poco escaso: este apartado puede poner en juego algunas algunas operaciones que no hemos visto en esta librería.
- El acabado de las figuras es algo rudimentario. en cambio, la librería Plotly me permite permitirte trazar figuras más profesionales, y una posibilidad sencilla es quizá importar los datos del archivo creado por el programa de map-reduce y representarlos gráficamente.
- La disponibilidad de datos de geolocalización puede permitirte alguna representación de la ubicación de los vehísulos registrados en su posición geográfica.

Estos ejemplos pueden servirte como pista, pero que no te limiten. Hay muchas otras posibilidades: geopandas, web scraping, etc.

En la evaluación, si este apartado está bien o muy bien, anota un 0,3 o 0,4. El 0,5 lo reservaremos para las situaciones en que se presente algo brillante, con alguna idea original o alguna técnica novedosa o complejidad especial o algún gráfico vistoso. Especialmente quien opta a un 9,5 o más, debe esmerarse en plantear este apartado a la altura de esa calificación.

g.1) Enunciado

A continuación, para practicar el empleo de la librería de *Scikit Learn*, se propone crear un modelo de regresión lineal con la finalidad de predecir el **salario** a partir de los datos proporcionados por las columnas en la variable *tabla_abreviada* (a excepción de la columna `salary_in_usd`). Para ello se propone seguir los siguientes pasos:

1. Realizar un análisis exploratorio de los datos disponibles y análisis de características.
2. Crear un modelo de regresión lineal para predecir el `salary_in_usd` en función del resto de características.
3. Aplicar el algoritmo PCA para estudiar la influencia de cada característica en la varianza explicada y estudiar la viabilidad de resolcer el problema usando menos variables.

Cabe destacar que el principal objetivo del ejercicio es utilizar la librería *Scikit Learn*, no elaborar un modelo preciso para resolver la tarea que se plantea.

Análisis exploratorio de los datos disponibles y análisis de características.

En primer lugar, para aplicar regresión, debemos trabajar con valores numéricos, por ello vamos a definir una función para traducir todas aquellas columnas cuyos valores son "categóricos" en el dataframe a enteros, para ello usaremos la función `pd.factorize()` de la librería *Pandas*:

In [52]:
```
# Function to encode categorical columns
def encode_categorical(data: pd.DataFrame, categorical_columns: list) -> pd.DataFra
    """
    Encode categorical columns in a DataFrame using factorization.

    Parameters
    ----------
        data (pd.DataFrame): The input DataFrame.
        categorical_columns (list): List of column names to encode.

    Returns
    ----------
        pd.DataFrame: A new DataFrame with encoded categorical columns.
    """
    for col in categorical_columns:
        data[col + '_encoded'], _ = pd.factorize(data[col])
    return data.drop(categorical_columns, axis=1)
```

Para cada columna especificada en `categorical_columns`, se aplica `pd.factorize()` para codificar los datos categóricos en enteros. Esto significa que asigna un entero único a cada categoría distinta dentro de la columna. La columna codificada se almacena como una nueva columna en el DataFrame con el nombre original de la columna al que se le agrega '_encoded'. Después de codificar todas las columnas especificadas, se devuelve un nuevo DataFrame con las columnas categóricas originales eliminadas.

Por otro lado, vamos a implementar una función para identificar y eliminar Outliers en el conjunto de datos que estamos analizando. Esta función aplica el método de **Z-score** para identificar Outliers.

El método consiste en aplicar **por columnas** la fórmula `(data - data.mean()) / data.std()`, dónde `data.mean()` y `data.std()` son la media y desviación típiva de la correspondiente columna.

Posteriormente, se establece el umbral por defecto para el método empleado que es el -3, 3, de manera que los datos cuyo z-score está por encima o debajo de ese valor se considerarán outliers (de nuevo, se realiza por columnas). Dichos elementos son eliminados.

In [53]:
```
# Function to perform outlier analysis
def outlier_analysis(data: pd.DataFrame) -> pd.DataFrame:
    """
    Identify and remove outliers in a DataFrame using z-scores.

    Parameters
    ----------
        data (pd.DataFrame): The input DataFrame.

    Returns
    ----------
        pd.DataFrame: A new DataFrame with outliers removed.
    """
    z_scores = (data - data.mean()) / data.std()
    outliers = (z_scores > 3) | (z_scores < -3)
    return data[~outliers.any(axis=1)]
```

Finalmente, la función `display_boxplots` se utiliza para visualizar y mostrar gráficos de caja (boxplots) de un DataFrame de pandas dado. Un boxplot es una representación gráfica de la distribución de un conjunto de datos. Proporciona un resumen visual de medidas estadísticas clave, como la mediana, los cuartiles y valores atípicos potenciales, lo que le permite comprender la dispersión y la tendencia central de los datos.

In [54]:
```
# Function to display boxplots
def display_boxplots(data: pd.DataFrame, title: str) -> None:
    """
    Display boxplots for the given DataFrame.

    Parameters
    ----------
        data (pd.DataFrame): The input DataFrame.
        title (str): Title for the boxplot.

    Returns
    ----------
        None
    """
    plt.figure(figsize=(10, 6))
```

```
    data.boxplot()
    plt.xticks(rotation=45)
    plt.title(title)
    plt.show()
```

En primer lugar, vamos a transformar todas las columnas categóricas en valores numéricos:

In [55]:
```
# Load the data
data_with_categorical = tabla_abreviada
data_with_categorical['employment_type'] = data_with_categorical['employment_type']

# Identify and encode categorical columns
categorical_columns = ['employment_type', 'job_title', 'employee_residence', 'compa
data_no_categorical = encode_categorical(data_with_categorical, categorical_columns

data_no_categorical
```

Out[55]:

	work_year	experience_level	salary_in_usd	remote_ratio	company_size	employment_t
0	2020	1	79833	0.0	3	
1	2020	3	260000	0.0	1	
2	2020	3	109024	50.0	2	
3	2020	1	20000	0.0	1	
4	2020	3	150000	50.0	3	
...	...	...	...	...	...	
602	2022	3	154000	100.0	2	
603	2022	3	126000	100.0	2	
604	2022	3	129000	0.0	2	
605	2022	3	150000	100.0	2	
606	2022	1	200000	100.0	3	

607 rows × 9 columns

Tras esto, consideramos que el año no debería tener relevancia en la predicción del salario y puede ser una variable que aporte ruido, es por ello que se decide prescindir de la misma:

In [56]:
```
# Drop 'working_year' column
data_without_working_year = data_no_categorical.drop('work_year', axis=1)
```

Ahora calculamos algunas estadísticas por columnas, para hacernos una idea de la distribución de las distintas variables:

In [57]:
```
# Display statistical information
print("Summary Statistics:")
```

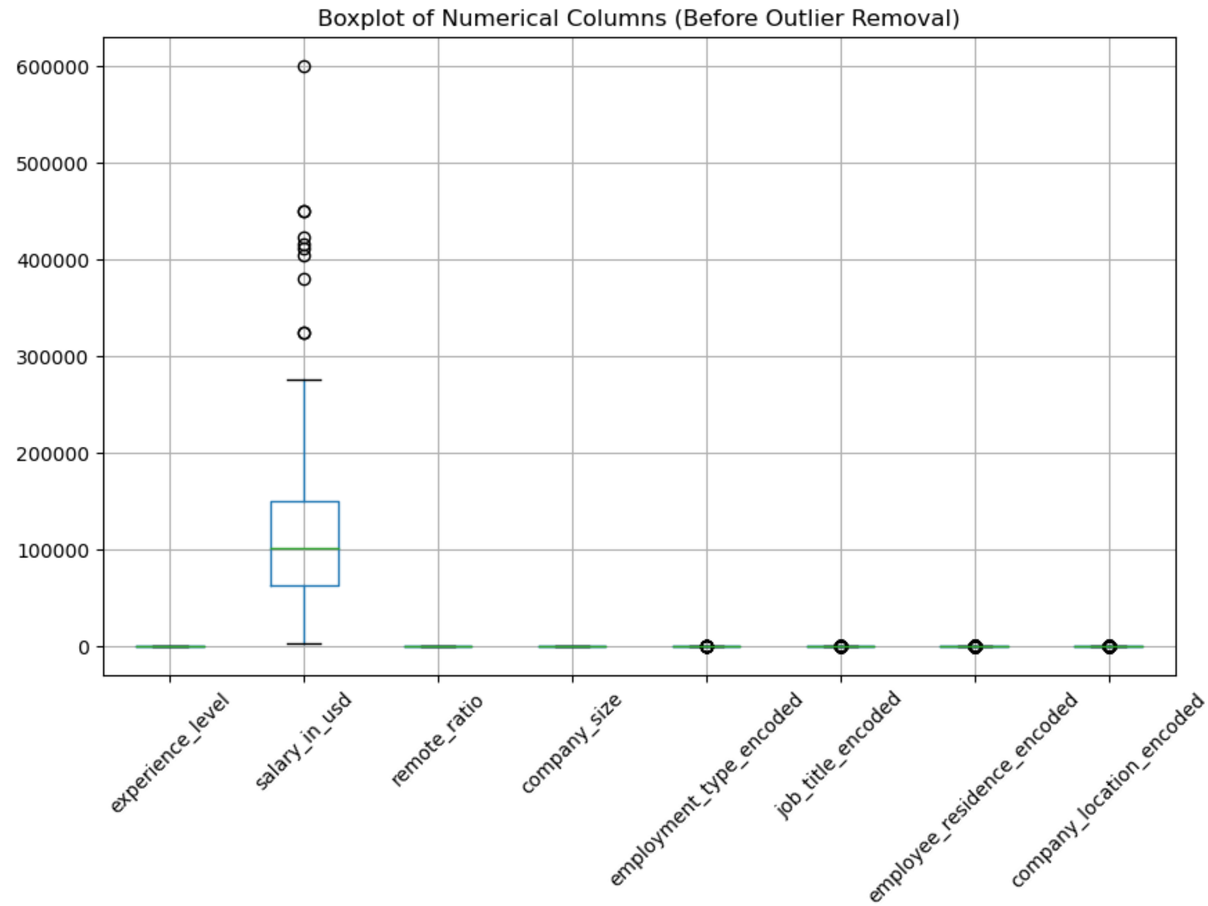
```python
print(data_without_working_year.describe())
```

```
Summary Statistics:
       experience_level  salary_in_usd  remote_ratio  company_size  \
count        607.000000     607.000000     607.00000    607.000000   
mean           1.820428  112297.869852      70.92257      2.189456   
std            1.167086   70957.259411      40.70913      0.654021   
min            0.000000    2859.000000       0.00000      1.000000   
25%            1.000000   62726.000000      50.00000      2.000000   
50%            2.000000  101570.000000     100.00000      2.000000   
75%            3.000000  150000.000000     100.00000      3.000000   
max            3.000000  600000.000000     100.00000      3.000000   

       employment_type_encoded  job_title_encoded  employee_residence_encoded  \
count               607.000000         607.000000                  607.000000   
mean                  0.060956          10.112026                    8.680395   
std                   0.360474          11.046734                   10.428762   
min                   0.000000           0.000000                    0.000000   
25%                   0.000000           2.000000                    4.000000   
50%                   0.000000           7.000000                    4.000000   
75%                   0.000000          11.500000                    8.000000   
max                   3.000000          49.000000                   56.000000   

       company_location_encoded  
count                607.000000  
mean                   7.645799  
std                    9.020589  
min                    0.000000  
25%                    4.000000  
50%                    4.000000  
75%                    8.000000  
max                   49.000000  
```

Cabe resaltar como en la columna de los salarios `salary_in_usd` el máximo es un valor mucho más elevado de lo que indican la media y los percentiles. Esto nos indica que podría haber presencia de Outliers, vamos a verlos en el siguiente gráfico:

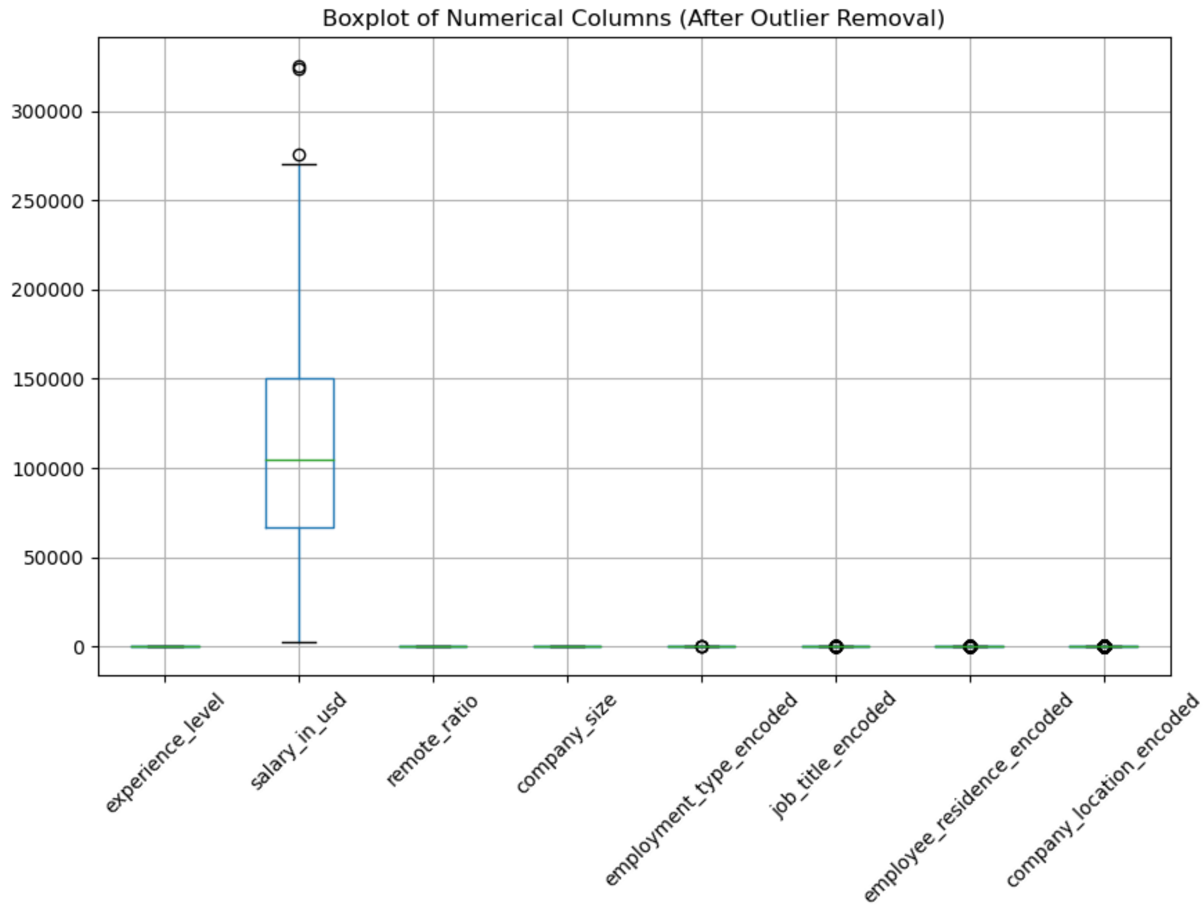
In [58]:

```python
# Display boxplots before removing outliers
display_boxplots(data_without_working_year, 'Boxplot of Numerical Columns (Before O
```



Como podemos observar, existen Outliers en la columna que deseamos predecir, es por ello que vamos a realizar un tratamiento de los mismos empleando la función descrita anteriormente. Vemos a continuación el resultado tras aplicar la función:

In [59]:
```
# Perform outlier analysis
data_after_outliers_analysis = outlier_analysis(data_without_working_year)

# Display boxplots after removing outliers
display_boxplots(data_after_outliers_analysis, 'Boxplot of Numerical Columns (After
```



Modelo de regresión lineal para predecir el `salary_in_usd` en función del resto de características.

Vamos a tratar de predecir el salario en dólares a partir del resto de variables disponibles. Para ello, vamos a usar la librería *Scikit Learn*.

En primer lugar hacemos la partición en conjunto de entrenamiento y test dejando un 80% de los datos para entrenamiento y el 20% restante para test. Usamos además una semilla para que los resultados sean reproducibles desde cualquier máquina.

In [60]:
```
X = data_after_outliers_analysis.drop(['salary_in_usd'], axis=1)  # Features
y = data_after_outliers_analysis['salary_in_usd']  # Target variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

En segundo lugar estandarizamos los datos. Esto implica transformar las características para que tengan una media de 0 y una desviación estándar de 1. La estandarización se utiliza a menudo cuando las características tienen diferentes unidades (como en este caso) o cuando el algoritmo es sensible a la escala de las características, como es el caso del descenso del gradiente (el que emplearemos). Así, por regla general es una buena práctica hacer esta transformación.

In [61]:
```
# Standardize the data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Ahora usamos la función `SGDRegressor()` de la librería *Scikit Learn* para construir el modelo (utiliza la función MSE como función de error y el método de gradiente desdendente), realizaremos unas 1000 iteraciones del algoritmo y evaluaremos el modelo con la métrica R^2, que nos indica el porcentaje de la varianza explicado por el modelo sobre la variable dependiente (en este caso el `salary_in_usd`).

In [62]:
```
# Linear Regression using all components

# Create an instance of the SGDRegressor with the desired hyperparameters
sgd_regressor = SGDRegressor(loss="squared_error", max_iter=1000, random_state=42)

# Fit the model to your data
sgd_regressor.fit(X_train_scaled, y_train)

# Evaluate the model on the test set
test_score = sgd_regressor.score(X_test_scaled, y_test)
print(f"Linear Regression R-squared on Test Set: {test_score}")
```

```
Linear Regression R-squared on Test Set: 0.29695050986545657
```

Como vemos, el resultado del coeficiente R^2 es pobre, lo que nos hace pensar que quizá un ajuste lineal no es la mejor alternativa para resolver el problema, no obstante, como se ha comentado, no es el objetivo de la práctica el obtener el mejor modelo posible para este cometido.

Aplicar el algoritmo PCA para estudiar la influencia de cada característica en la varianza explicada y estudiar la viabilidad de resolcer el problema usando menos variables.

A continuación usaremos el algoritmo *PCA* de la librería de *Scikit Learn* Para ver la influencia de cada variable independiente sobre la variable que pretendemos predecir, esto nos ayudará eliminar variables que no tengan un efecto notable en este cometido simplificando el problema y consiguiendo resultados similares con un modelo más sencillo.

Usaremos la función `PCA()` de la librería, y vemos las 6 variables con mayor influencia en la varianza de la variable dependiente:

In [63]:
```
# Apply PCA
pca = PCA()
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)

# Results: Explained Variance Ratio by each Principal Component
explained_variance = pca.explained_variance_ratio_
```

```
column_names = X.columns

# Selecting the top components
num_components = 6  # Selecting the top 6 components
top_components = pd.Series(explained_variance, index=column_names).nlargest(num_com

# Get the indices of the most important components
most_important_indices = np.argsort(pca.explained_variance_)[::-1][:num_components]

print("Top Components based on Explained Variance:")
print(top_components)
print ("Cumulative variance explained: ")
print(top_components.sum())
print("Indices of most important components: ")
print(most_important_indices)
```

```
Top Components based on Explained Variance:
experience_level              0.265508
remote_ratio                  0.169384
company_size                  0.155172
employment_type_encoded       0.140609
job_title_encoded             0.125013
employee_residence_encoded    0.117754
dtype: float64
Cumulative variance explained:
0.9734404997956684
Indices of most important components:
[0 1 2 3 4 5]
```

Como vemos, la que más influencia tiene es el nivel de experiencia del empleado, algo que es bastante intuitivo pues cuanto mayor sea la experiencia en el sector de un empleado, normalmente su salario es mayor también. Cabe destacar cómo el segundo factor es el ratio de porcentaje remoto del puesto y la tercera el tamaño de la compañía. En total, la varianza explicada acumulada de la variable dependiente con estas 6 componentes es casi del 100 (un 97 aproximadamente), por lo que podemos considerar eliminar la variable `company_location` del estudio y tratar de entrenar el modelo con las demás.

Vamos pues a eliminar de nuestro conjunto de entrenamiento la última variable y a reentrenar el modelo de nuevo:

In [64]:
```
# Now PCA using only the selected components for training and testing data
X_train_pca = X_train_pca[:, most_important_indices]
X_test_pca = X_test_pca[:, most_important_indices]

# Linear Regression using TOP 5 selected PCA components
# Create an instance of the SGDRegressor with the desired hyperparameters
sgd_regressor_pca = SGDRegressor(loss="squared_error", max_iter=1000, random_state=

# Fit the model to your data
sgd_regressor_pca.fit(X_train_pca, y_train)

# Evaluate the model on the test set
```

```
test_score_pca = sgd_regressor_pca.score(X_test_pca, y_test)
print(f"Linear Regression R-squared on Test Set with selected components: {test_sco
```

```
Linear Regression R-squared on Test Set with selected components: 0.3031308784949323
7
```

Como vemos el valor de R^2 es casi idéntico al caso con todas las variables (incluso ligeramente superior). Con esto hemos conseguido reducir la complejidad del problema y conseguir un modelo con un rendimiento similar al caso con todas las variables.

A continuación se presentan una serie de tests unitarios para poder comprobar los resultados, para ello empleamos la librería *pytest* en su versión interactiva llamada *ipytest*:

In [65]:
```
! pip install ipytest
```

```
Requirement already satisfied: ipytest in c:\users\pablo\anaconda3\lib\site-packages
(0.13.3)
Requirement already satisfied: ipython in c:\users\pablo\anaconda3\lib\site-packages
(from ipytest) (8.15.0)
Requirement already satisfied: packaging in c:\users\pablo\anaconda3\lib\site-packag
es (from ipytest) (23.1)
Requirement already satisfied: pytest>=5.4 in c:\users\pablo\anaconda3\lib\site-pack
ages (from ipytest) (7.4.0)
Requirement already satisfied: iniconfig in c:\users\pablo\anaconda3\lib\site-packag
es (from pytest>=5.4->ipytest) (1.1.1)
Requirement already satisfied: pluggy<2.0,>=0.12 in c:\users\pablo\anaconda3\lib\sit
e-packages (from pytest>=5.4->ipytest) (1.0.0)
Requirement already satisfied: colorama in c:\users\pablo\anaconda3\lib\site-package
s (from pytest>=5.4->ipytest) (0.4.6)
Requirement already satisfied: backcall in c:\users\pablo\anaconda3\lib\site-package
s (from ipython->ipytest) (0.2.0)
Requirement already satisfied: decorator in c:\users\pablo\anaconda3\lib\site-packag
es (from ipython->ipytest) (5.1.1)
Requirement already satisfied: jedi>=0.16 in c:\users\pablo\anaconda3\lib\site-packa
ges (from ipython->ipytest) (0.18.1)
Requirement already satisfied: matplotlib-inline in c:\users\pablo\anaconda3\lib\sit
e-packages (from ipython->ipytest) (0.1.6)
Requirement already satisfied: pickleshare in c:\users\pablo\anaconda3\lib\site-pack
ages (from ipython->ipytest) (0.7.5)
Requirement already satisfied: prompt-toolkit!=3.0.37,<3.1.0,>=3.0.30 in c:\users\pa
blo\anaconda3\lib\site-packages (from ipython->ipytest) (3.0.36)
Requirement already satisfied: pygments>=2.4.0 in c:\users\pablo\anaconda3\lib\site-
packages (from ipython->ipytest) (2.15.1)
Requirement already satisfied: stack-data in c:\users\pablo\anaconda3\lib\site-packa
ges (from ipython->ipytest) (0.2.0)
Requirement already satisfied: traitlets>=5 in c:\users\pablo\anaconda3\lib\site-pac
kages (from ipython->ipytest) (5.7.1)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in c:\users\pablo\anaconda3\lib\s
ite-packages (from jedi>=0.16->ipython->ipytest) (0.8.3)
Requirement already satisfied: wcwidth in c:\users\pablo\anaconda3\lib\site-packages
(from prompt-toolkit!=3.0.37,<3.1.0,>=3.0.30->ipython->ipytest) (0.2.5)
Requirement already satisfied: executing in c:\users\pablo\anaconda3\lib\site-packag
es (from stack-data->ipython->ipytest) (0.8.3)
Requirement already satisfied: asttokens in c:\users\pablo\anaconda3\lib\site-packag
es (from stack-data->ipython->ipytest) (2.0.5)
Requirement already satisfied: pure-eval in c:\users\pablo\anaconda3\lib\site-packag
es (from stack-data->ipython->ipytest) (0.2.2)
Requirement already satisfied: six in c:\users\pablo\anaconda3\lib\site-packages (fr
om asttokens->stack-data->ipython->ipytest) (1.16.0)
```

In [66]:

```python
ipytest.autoconfig()
```

In [67]:

```python
# Pruebas de funcionamiento, también tarea del estudiante:

def test_dataframe_has_categorical_columns():
    # Get a list of column names with categorical data type
    categorical_columns = data_with_categorical.select_dtypes(include=['category'])
    # Check if there are categorical columns
    assert len(categorical_columns) > 0

def test_dataframe_has_not_categorical_columns():
```

```python
    # Get a list of column names with categorical data type
    categorical_columns = data_no_categorical.select_dtypes(include=['category']).c
    # Check there are no categorical columns
    assert len(categorical_columns) == 0

def test_no_working_year_column():
    # Check if 'working_year' column is not present in the DataFrame
    assert 'working_year' not in data_without_working_year.columns

def test_outliers_analysis_function():
    # Store the DataFrame before applying outlier analysis
    df_before_outliers = data_without_working_year
    # Apply the outlier analysis function
    df_after_outliers = outlier_analysis(df_before_outliers)
    # Check the number of rows before and after the analysis
    assert len(df_before_outliers) > len(df_after_outliers)
    assert len(df_after_outliers) == 547
    assert len(df_before_outliers) == 607

def test_datasets_lengths():
    # Check the length of the training and test datasets
    assert len(X_train) == 437
    assert len(X_test) == 110

def test_standarization():
    tolerance = 1e-9
    # Check if the mean and standard deviation of the scaled training data are clos
    assert np.allclose(X_train_scaled.mean(), 0.0, rtol=tolerance, atol=tolerance)
    assert np.allclose(X_train_scaled.std(), 1.0, rtol=tolerance, atol=tolerance)

def test_most_important_indices():
    # Define a gold list of important indices
    gold_list = [0, 1, 2, 3, 4, 5]
    # Check if the calculated list of most important indices matches the gold list
    assert gold_list == most_important_indices.tolist()

def test_scores_before_and_after_pca():
    tolerance = 1e-9
    # Check if test scores before and after PCA are within a specific tolerance
    assert np.allclose(round(test_score, 3), 0.297, rtol=tolerance, atol=tolerance)
    assert np.allclose(round(test_score_pca, 3), 0.303, rtol=tolerance, atol=tolera
    # Check if test score after PCA is higher than the test score before PCA
    assert test_score_pca > test_score

ipytest.run("-v")
```

```
======================================= test session starts =========================
===============
platform win32 -- Python 3.11.5, pytest-7.4.0, pluggy-1.0.0
rootdir: c:\Users\pablo\OneDrive\Documentos\GitHub\MasterBigDataML-PythonCourse\pyth
on project\-- data_science_salaries
plugins: anyio-3.5.0
collected 8 items

t_240cd1b75e864cf49072bbdeaea645a5.py ........
[100%]

======================================== 8 passed in 0.04s ==========================
===============
```

Out[67]: <ExitCode.OK: 0>

Datos personales

- **Apellidos:** Borrego Megías
- **Nombre:** Alejandro
- **Email:** alejbormeg@gmail.com
- **Fecha:** 19/10/2023

Ficha de autoevaluación

Apartado	Calificación	Comentario
a)	2.0 / 2.0	Me ha parecido un ejercicio interesante para el manejo de ficheros en formato '.csv'
b)	2.0 / 2.0	Buen ejercicio para el manejo de diccionarios convencionales y por defecto leyendo de ficheros
c)	2.0 / 2.0	Ejercicio con complejidad para el manejo de diccionario y la preparación de datos
d)	1.5 / 1.5	Me ha parecido un ejercicio sencillo para usar la librería matplotlib
e)	1.5 / 1.5	Buen ejercicio para dar a conocer los dataframes de pandas y sus ventajas para manipular datos y ficheros '.csv'
f)	0.5 / 0.5	Buena oportunidad para practicar la técnica Map-reduce
g)	0.5 / 0.5	Creo que he podido elaborar un buen ejercicio para practicar con `sklearn` y crear una batería de tests unitarios
Total	**10.0** / 10.0	**Sobresaliente**

Ayuda recibida y fuentes utilizadas

Las fuentes principales empleadas para la elaboración del trabajo han sido las siguientes (son enlaces a los sitios web de la documentación oficial):

- PEP 8 Style Guide
- Librería csv
- Numpy
- Pandas
- PCA
- Scikit-Learn Linear Regression
- Libreria ipytest
- Matplotlib
- Python Collections Dictionaries
- Apuntes del módulo y material audiovisual para el ejercicio de Map-Reduce

Comentario adicional

La práctica ha sido complicada de realizar por la gran cantidad de iteraciones que han sido necesarias para aclarar algunos enunciados confusos y eliminar erratas. No obstante, a su vez esta iteración con el profesor me ha hecho experimentar una mayor similitud con el desarrollo de proyectos software en el mundo laboral.

In [68]:
```
# Esta celda se ha de respetar: está aquí para comprobar el funcionamiento de algun
```