EDA

EDA atau "Exploratory Data Analysis" merupakan suatu proses dalam analisis data yang digunakan untuk memahami dataset yang tersedia. Tujuannya adalah untuk menggali informasi dan wawasan yang tersembunyi dalam data, sehingga dapat membantu peneliti atau analis dalam membuat keputusan yang lebih baik.

EDA melibatkan penggunaan teknik statistik dan visualisasi data untuk memahami pola, tren, dan hubungan antara variabel dalam dataset. Beberapa teknik yang umum digunakan dalam EDA meliputi histogram, scatter plot, box plot, dan heat map. Selain itu, EDA juga dapat melibatkan analisis deskriptif, seperti mean, median, modus, dan deviasi standar, serta pengujian hipotesis.

EDA sangat penting dalam data science karena dapat membantu mengidentifikasi masalah atau anomali dalam data, menentukan apakah data bersih atau tidak, dan menentukan apakah model atau analisis yang diusulkan tepat atau tidak. Dengan demikian, EDA dapat membantu meningkatkan kualitas dan akurasi analisis data dan mengurangi risiko kesalahan interpretasi.

Pada kasus ini, akan dilakukan tahap EDA dari dataset yang telah diberikan yaitu mengenai "Ask A Manager Salary Survey 2021 (Responses)"

> Langkah pertama yang perlu dilakukan adalah melakukan import library. Berikut merupakan penjelasan mengenai library yang digunakan :

1. Pandas : library yang digunakan untuk manipulasi dan analisis data.Ini menyediakan struktur data dan fungsi yang diperlukan untuk bekerja dengan data terstruktur secara mulus.
2. Numpy : library yang digunakan untuk perhitungan numerik dengan Python. Ini memberikan dukungan untuk array multi-dimensi dan fungsi matematika untuk beroperasi pada array ini.
3. Seaborn : library untuk visualisasi data berdasarkan matplotlib. Ini menyediakan antarmuka tingkat tinggi untuk membuat grafik statistik yang informatif dan menarik.
4. Matplotlib.pyplot : sub-library matplotlib yang digunakan untuk membuat berbagai jenis plot seperti plot garis, plot pencar, plot batang, histogram, dll

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

> Fungsi "pd.read_excel()" adalah fungsi panda yang membaca file Excel dan membuat objek DataFrame dari datanya. Argumen untuk fungsi tersebut adalah jalur file atau URL ke file Excel. Dalam hal ini, jalur file adalah "Ask A Manager Salary Survey 2021 (Responses).xlsx".

> Setelah kode dieksekusi, DataFrame "df" akan berisi data dari file Excel, yang kemudian dapat dimanipulasi dan dianalisis menggunakan panda dan pustaka Python lainnya.

```
df = pd.read_excel("Ask A Manager Salary Survey 2021 (Responses).xlsx")
```

Selanjutnya kita dapat melihat 5 data teratas dari dataframe

```
df.head(5)
```

Timestamp	How old are you?	What industry do you work in?	Job title	If your job title needs additional context, please clarify here:	What is your annual salary? (You'll indicate the currency in a later question. If you are part-time or hourly, please enter an equivalent	How much additional monetary compensation do you get, if any (for example, bonuses or overtime in an average year. Only include	i
						the job 40	benefits,

Dapat dilihat bahwa dataframe memiliki dimensi yaitu 27946 row dengan 18 column.

```
df.shape
```

```
(27946, 18)
```

Dikarenakan nama column yang sebelumnya terlalu panjang dan kompleks, jadi disini akan dilakukan untuk perubahan nama column agar lebih simple.

```
df = pd.DataFrame(df.values , columns=(["time",
                                        "age",
                                        "industry",
                                        "job_title",
                                        "need_additional_context",
                                        "salary",
                                        "monetary_compentation",
                                        "currency",
                                        "indicate_currency",
                                        "income_additional_contex",
                                        "country",
                                        "US_state",
                                        "city",
                                        "work_experience_overall",
                                        "work_experince_field",
                                        "hights_education_complate",
                                        "gender",
                                        "your_race"]) , index= range(0,len(df)))
```

Setelah itu kita dapat melihat bahwa nama column telah berubah.

```
df.head(5)
```

	time	age	industry	job_title	need_additional_context	salary	monetar
0	2021-04-27 11:02:09.743	25-34	Education (Higher Education)	Research and Instruction Librarian	NaN	55000.0	
1	2021-04-27 11:02:21.562	25-34	Computing or Tech	Change & Internal Communications Manager	NaN	54600.0	
2	2021-04-27 11:02:38.125	25-34	Accounting, Banking & Finance	Marketing Specialist	NaN	34000.0	
3	2021-04-27 11:02:40.643	25-34	Nonprofits	Program Manager	NaN	62000.0	
4	2021-04-27 11:02:41.793	25-34	Accounting, Banking & Finance	Accounting Manager	NaN	60000.0	

CLEANSING

> Tahap cleansing atau pembersihan data adalah tahap dalam analisis data yang bertujuan untuk membersihkan data dari nilai yang hilang, duplikat, atau tidak akurat. Tahap cleansing sangat penting dalam analisis data karena data yang tidak bersih atau tidak teratur dapat menyebabkan kesalahan dalam analisis dan menyebabkan hasil yang tidak dapat diandalkan.

> Tahap cleansing melibatkan langkah-langkah seperti:

1. Identifikasi nilai yang hilang: Menemukan nilai yang hilang dan menentukan apakah data yang hilang tersebut dapat diisi dengan nilai yang masuk akal atau harus dihapus.
2. Identifikasi data duplikat: Mencari data yang duplikat dan menentukan apakah duplikat tersebut harus dihapus atau tidak.
3. Menangani data yang tidak valid: Menemukan data yang tidak valid atau tidak masuk akal dan menentukan apakah data tersebut harus dihapus atau dikoreksi.

> Setelah data dibersihkan, langkah berikutnya adalah melakukan eksplorasi data dan analisis data. Tahap cleansing yang benar dan menyeluruh sangat penting untuk mendapatkan hasil analisis yang akurat dan dapat diandalkan.

```
df.isna().sum()
#data.info()
```

```
time                            0
age                             0
industry                       72
job_title                       0
need_additional_context     20713
salary                          0
monetary_compentation        7256
currency                        0
indicate_currency           27749
income_additional_contex    24912
country                         0
US_state                     4982
city                           75
work_experience_overall         0
work_experince_field            0
hights_education_complate     214
gender                        167
your_race                     169
dtype: int64
```

CLEANSING INDUSTRY

Disini kita akan melihat data dimana column industry memiliki nilai Null. Dapat diketahui bahwa terdapat 72 data yg kosong.

```
datax = df[(df["industry"].isna()==True)]
datax
#len(data)
#72 data industri kosong
```

	time	age	industry	job_title	need_additional_context	salary	m
360	2021-04-27 11:08:41.197	45-54	NaN	Proposal Manager	NaN	87938.0	
645	2021-04-27 11:12:58.989	35-44	NaN	Legal editor	NaN	82000.0	
1604	2021-04-27 11:28:19.790	25-34	NaN	Commissioning Editor	NaN	36000.0	
2055	2021-04-27 11:37:01.190	35-44	NaN	Research Manager	NaN	115000.0	
2063	2021-04-27 11:37:11.960	35-44	NaN	office manager	manage a building vs managing employees	40000.0	
...	...	...	...	...	...	...	
27250	2021-10-26 13:46:24.184	55-64	NaN	Technician Automation	NaN	35000.0	
27592	2021-12-28 18:15:36.647	45-54	NaN	Associate professor	Tenured	80000.0	

Karna sebelumnya terdapat data row dimana column industry Null. Maka kita akan membuang data tersebut dengan menggunakan method dropna dengan paramater subset = nama_column.

2022-08-04 18-

```
df = df.dropna(subset=["industry"])
df.isna().sum()
```

```
time                            0
age                             0
industry                        0
job_title                       0
need_additional_context     20653
salary                          0
monetary_compentation        7232
currency                        0
indicate_currency           27678
income_additional_contex    24842
country                         0
US_state                     4966
city                           75
work_experience_overall         0
work_experince_field            0
hights_education_complate     211
gender                        166
your_race                     168
dtype: int64
```

```
industri = df["industry"].unique()
len(industri)
```

```
#terdapat 1209 nama industri
```

```
1209
```

CLEANSING SALARY I (CONVERT BY CURRENCY)

> Disini kita akan mengubah type data dari column "salary" yang bermula bertipe object menjadi tipe data integer dengan menggunakan method astype.

```
print(df["salary"].dtypes)
df["salary"] = df["salary"].astype(int)
print(df["salary"].dtypes)
#rubah tipe salary ke int
```

```
object
int32
```

> Selanjutnya kita mengecek value dari data column currency.

```
df["currency"].unique()
#check tipe currency
```

```
array(['USD', 'GBP', 'CAD', 'EUR', 'AUD/NZD', 'Other', 'CHF', 'ZAR',
       'SEK', 'HKD', 'JPY'], dtype=object)
```

> Setelah itu kita mengecek juga value dari data column indicate currency.

```
df["indicate_currency"].unique()
#check tipe indicate currency
```

```
array([nan, 'INR', 'Peso Argentino', 76302.34,
       'My bonus is based on performance up to 10% of salary',
       'I work for an online state university, managing admissions data. Not direct
tech support. ',
       0.0, 'MYR', 'CHF', 'KWD', 'NOK', 'Na ', 'USD', 'BR$', 'SEK',
       'Base plus Commission ', 'canadian', 'Dkk', 'EUR', 'COP', 'TTD',
       'Indian rupees', 'BRL (R$)', 'Mexican pesos', 'CZK', 'GBP', 'DKK',
       'Bdt', 'RSU / equity', 'ZAR', 'Additonal = Bonus plus stock',
       'American Dollars', 'Php', 'PLN (Polish zloty)',
       'Overtime (about 5 hours a week) and bonus', 'czech crowns',
       'Stock ', 'TRY', 'Norwegian kroner (NOK)', 'CNY', 'ILS/NIS',
       55000.0, 'AUD & NZD are not the same currency...', 'US Dollar',
       'Canadian ', 'AUD', 'BRL', 'NIS (new Israeli shekel)', '-',
```

```
       'RMB (chinese yuan)', 'Taiwanese dollars',
       "AUD and NZD aren't the same currency, and have absolutely nothing to do with
each other :(",
       'NZD', 'Philippine Peso', 'SGD', 'KRW (Korean Won)', 'Czk', 'THB',
       'IDR ', 'Sgd', 'Nok', 'ILS (Shekel)',
       '6000 in stock grants annually', 'DKK ', 'China RMB',
       'AUD Australian ', 'LKR', 'Polish Złoty', 'Philippine peso (PHP)',
       'Australian Dollars ', 'PHP',
       'Many non-salary benefits - travel, free healthcare for self, very low for
family, non-taxable housing allowance ',
       'Equity',
       'It’s marketed as £22000 but we get paid pro-rats, so no pay for the school
holidays.',
       'additional compensation is for overtime (i am paid hourly) so it varies. i have
included an estimate',
       'ARS', 'Argentinian peso (ARS)', 'Israeli Shekels', 'ILS', 'MXN',
       'PhP (Philippine Peso)',
       'Converted mine into USD for your easyness', 'PLN', 'KRW', 'SAR',
       'RM', 'IDR', 'None', 'Argentine Peso', 'Philippine Pesos', 'Ils',
       'Rs', 'INR (Indian Rupee)', 'NTD', 'Danish Kroner', 'CAD',
       'Korean Won ', 'dkk', 'Euro', 'SGD ', 'Mexican Pesos',
       'THAI  BAHT',
       'Option to get 2x or 1.5x if taking on a weekend day in the summer',
       'Thai Baht ', 47000.0, 'na', 'Canadian', 'N/a',
       'up to 12% annual bonus', 'croatian kuna', 'PLN (Zwoty)', 5.0,
       'Rupees', 'Singapore Dollara', 'NGN'], dtype=object)
```

> Setelah dicek ternyata terdapat beberapa data dimana variable currency diisi nilai other dan indicate_currency tidak diberi nilai ('Nan'). Oleh karena itu, kita akan menginisialisasi beberapa row data untuk data currency dengan melihat negara tempat mereka bekerja.

```
df.loc[9344, 'indicate_currency'] = 'USD'
df.loc[18843, 'indicate_currency'] = 'MYR'
df.loc[18882, 'indicate_currency'] = 'MYR'
df.loc[18904, 'indicate_currency'] = 'USD'
```

> Setelah itu kita akan mengecek data currency dengan nilai other dan indicate currency bernilai ('Nan'). Hal ini diperlukan agar dapat mengkonversi data salary berdasarkan currency/indicate currency.

```
datax = df[(df["currency"] == "Other") & (df["indicate_currency"].isna() == True)]
len(datax)
#check data currency dengan nilai other dan indicate currency nilainya kosong
#hasil nya 0 berarti semua data terisi
```

```
0
```

> Selanjutnya kita cek juga data currency dengan nilai "USD" dan indicate currency yang bernilai "USD" dan dapat diketahui bahwa terdapat 23241 data.

```
hasil = df[(df["currency"] == 'USD') | (df["indicate_currency"] == 'USD')]
len(hasil)
```

```
23241
```

> Kemudian kita akan mengecek data currency dengan nilai other dan indicate currency != "USD". Hal ini akan dilakukan dalam perbaikan penulisan format mata uang agar bisa dikonversi ke USD.

```
hasil2 = df[(df["currency"] == 'Other') & (df["indicate_currency"] != 'USD')]
hasil2["indicate_currency"].unique()
```

```
array(['INR', 'Peso Argentino', 'MYR', 'CHF', 'NOK', 'BR$', 'SEK', 'Dkk',
       'EUR', 'TTD', 'Indian rupees', 'BRL (R$)', 'Mexican pesos', 'CZK',
       'GBP', 'DKK', 'Bdt', 'ZAR', 'American Dollars', 'Php',
       'PLN (Polish zloty)', 'czech crowns', 'TRY',
       'Norwegian kroner (NOK)', 'CNY', 'ILS/NIS', 'US Dollar', 'BRL',
       'NIS (new Israeli shekel)', 'RMB (chinese yuan)', 'AUD',
       'Taiwanese dollars', 'Philippine Peso', 'SGD', 'KRW (Korean Won)',
       'Czk', 'THB', 'IDR ', 'Sgd', 'Nok', 'NZD', 'ILS (Shekel)', 'DKK ',
       'China RMB', 'AUD Australian ', 'LKR', 'Polish Złoty',
       'Philippine peso (PHP)', 'Australian Dollars ', 'PHP', 'Equity',
       'ARS', 'Argentinian peso (ARS)', 'Israeli Shekels', 'ILS', 'MXN',
       'PhP (Philippine Peso)', 'PLN', 'KRW', 'SAR', 'RM', 'IDR',
       'Argentine Peso', 'Philippine Pesos', 'Ils', 'INR (Indian Rupee)',
       'NTD', 'Danish Kroner', 'CAD', 'Korean Won ', 'dkk', 'Euro',
       'SGD ', 'Mexican Pesos', 'THAI  BAHT', 'Thai Baht ',
       'croatian kuna', 'PLN (Zwoty)', 'Rupees', 'Singapore Dollara',
       'NGN'], dtype=object)
```

> Dikarenakan banyak data indicate currency yang tidak sesuai dengan format mata uang negara. Maka kita melakukan inisialisasi data dengan menggunakan method replace.

```
df['currency'] = df['currency'].replace('AUD/NZD', 'AUD')
#df['indicate_currency'] = df['indicate_currency'].replace('Israeli Shekels', 'ILS')
df['indicate_currency'] = df['indicate_currency'].replace('Php', 'PHP')
df['indicate_currency'] = df['indicate_currency'].replace('Equity', 'EUR')
df['indicate_currency'] = df['indicate_currency'].replace('RM', 'MYR')
df['indicate_currency'] = df['indicate_currency'].replace('Australian Dollars', 'AUD')
```

```
df['indicate_currency'] = df['indicate_currency'].replace('czech crowns', 'AUD')
df['indicate_currency'] = df['indicate_currency'].replace('ILS (Shekel)', 'ILS')
df['indicate_currency'] = df['indicate_currency'].replace('Taiwanese dollars', 'TWD')
df['indicate_currency'] = df['indicate_currency'].replace('Rupees', 'INR')
df['indicate_currency'] = df['indicate_currency'].replace('croatian kuna', 'HRK')
df['indicate_currency'] = df['indicate_currency'].replace('Singapore Dollara', 'SGD')
df['indicate_currency'] = df['indicate_currency'].replace('Peso Argentino', 'ARS')
df['indicate_currency'] = df['indicate_currency'].replace('BR$', 'BRL')
df['indicate_currency'] = df['indicate_currency'].replace('BRL (R$)', 'BRL')
df['indicate_currency'] = df['indicate_currency'].replace('PLN (Polish zloty)', 'PLN')
df['indicate_currency'] = df['indicate_currency'].replace('ILS/NIS', 'ILS')
df['indicate_currency'] = df['indicate_currency'].replace('US Dollar', 'USD')
df['indicate_currency'] = df['indicate_currency'].replace('Philippine peso (PHP)', 'PHP')
df['indicate_currency'] = df['indicate_currency'].replace('Philippine Pesos', 'PHP')
df['indicate_currency'] = df['indicate_currency'].replace('RMB (chinese yuan)', 'CNY')
df['indicate_currency'] = df['indicate_currency'].replace('KRW (Korean Won)', 'KRW')
df['indicate_currency'] = df['indicate_currency'].replace('Norwegian kroner (NOK)', 'NOK')
df['indicate_currency'] = df['indicate_currency'].replace('Mexican pesos', 'MXN')
df['indicate_currency'] = df['indicate_currency'].replace('Indian rupees', 'INR')
df['indicate_currency'] = df['indicate_currency'].replace('NIS (new Israeli shekel)', 'ILS')
df['indicate_currency'] = df['indicate_currency'].replace('INR (Indian Rupee)', 'INR')
df['indicate_currency'] = df['indicate_currency'].replace('Israeli Shekels', 'ILS')
df['indicate_currency'] = df['indicate_currency'].replace('Korean Won ', 'KRW')
df['indicate_currency'] = df['indicate_currency'].replace('Danish Kroner', 'DKK')
df['indicate_currency'] = df['indicate_currency'].replace('Dkk', 'DKK')
df['indicate_currency'] = df['indicate_currency'].replace('Bdt', 'BDT')
df['indicate_currency'] = df['indicate_currency'].replace('dkk', 'DKK')
df['indicate_currency'] = df['indicate_currency'].replace('Czk', 'Czk')
df['indicate_currency'] = df['indicate_currency'].replace('ils', 'ILS')
df['indicate_currency'] = df['indicate_currency'].replace('THAI BAHT', 'THB')
df['indicate_currency'] = df['indicate_currency'].replace('Thai Baht ', 'THB')

df['indicate_currency'] = df['indicate_currency'].replace('Polish Złoty', 'PHP')
df['indicate_currency'] = df['indicate_currency'].replace('American Dollars', 'USD')
df['indicate_currency'] = df['indicate_currency'].replace('Philippine Peso', 'PHP')
df['indicate_currency'] = df['indicate_currency'].replace('China RMB', 'CNY')
df['indicate_currency'] = df['indicate_currency'].replace('Australian Dollars ', 'AUD')
df['indicate_currency'] = df['indicate_currency'].replace('Argentinian peso (ARS)', 'ARS')
df['indicate_currency'] = df['indicate_currency'].replace('Philippine Peso', 'PHP')
df['indicate_currency'] = df['indicate_currency'].replace('PhP (Philippine Peso)', 'PHP')

df['indicate_currency'] = df['indicate_currency'].replace('Ils', 'ILS')
df['indicate_currency'] = df['indicate_currency'].replace('Mexican Pesos', 'MXN')
df['indicate_currency'] = df['indicate_currency'].replace('PLN (Zwoty)', 'PLN')
df['indicate_currency'] = df['indicate_currency'].replace('THAI  BAHT', 'THB')
df['indicate_currency'] = df['indicate_currency'].replace('AUD Australian ', 'AUD')
df['indicate_currency'] = df['indicate_currency'].replace('Argentine Peso', 'ARS')
df['indicate_currency'] = df['indicate_currency'].replace('Czk', 'ARS')
df['indicate_currency'] = df['indicate_currency'].replace('Sgd', 'SGD')
df['indicate_currency'] = df['indicate_currency'].replace('Nok', 'SGD')
df['indicate_currency'] = df['indicate_currency'].replace('Euro', 'EUR')
```

```
df['indicate_currency'] = df['indicate_currency'].replace('IDR ', 'IDR')
df['indicate_currency'] = df['indicate_currency'].replace('DKK ', 'DKK')
df['indicate_currency'] = df['indicate_currency'].replace('AUD & NZD are not the same currenc
df['indicate_currency'] = df['indicate_currency'].replace('Converted mine into USD for your e
df['indicate_currency'] = df['indicate_currency'].replace('Canadian ', 'CAD')
df['indicate_currency'] = df['indicate_currency'].replace('Canadian', 'CAD')
df['indicate_currency'] = df['indicate_currency'].replace('canadian', 'CAD')
df['indicate_currency'] = df['indicate_currency'].replace('SGD ', 'SGD')
df['indicate_currency'] = df['indicate_currency'].replace(76302.34, np.nan)
df['indicate_currency'] = df['indicate_currency'].replace(0, np.nan)
df['indicate_currency'] = df['indicate_currency'].replace('My bonus is based on performance u
df['indicate_currency'] = df['indicate_currency'].replace('I work for an online state univers
df['indicate_currency'] = df['indicate_currency'].replace('Base plus Commission ', np.nan)
df['indicate_currency'] = df['indicate_currency'].replace('Additonal = Bonus plus stock', np.
df['indicate_currency'] = df['indicate_currency'].replace('RSU / equity', np.nan)
df['indicate_currency'] = df['indicate_currency'].replace("AUD and NZD aren't the same curren
df['indicate_currency'] = df['indicate_currency'].replace('6000 in stock grants annually', np
df['indicate_currency'] = df['indicate_currency'].replace('Many non-salary benefits - travel,
df['indicate_currency'] = df['indicate_currency'].replace('It’s marketed as £22000 but we get
df['indicate_currency'] = df['indicate_currency'].replace('additional compensation is for ove
df['indicate_currency'] = df['indicate_currency'].replace('None', np.nan)
df['indicate_currency'] = df['indicate_currency'].replace('Rs', np.nan)
df['indicate_currency'] = df['indicate_currency'].replace('na', np.nan)
df['indicate_currency'] = df['indicate_currency'].replace('N/a', np.nan)
df['indicate_currency'] = df['indicate_currency'].replace('up to 12% annual bonus', np.nan)
df['indicate_currency'] = df['indicate_currency'].replace('Overtime (about 5 hours a week) an
df['indicate_currency'] = df['indicate_currency'].replace('Stock ', np.nan)
df['indicate_currency'] = df['indicate_currency'].replace('Na ', np.nan)
df['indicate_currency'] = df['indicate_currency'].replace('-', np.nan)
df['indicate_currency'] = df['indicate_currency'].replace(5, np.nan)
df['indicate_currency'] = df['indicate_currency'].replace(47000, np.nan)
df['indicate_currency'] = df['indicate_currency'].replace(55000.0, np.nan)
df['indicate_currency'] = df['indicate_currency'].replace('Option to get 2x or 1.5x if taking
```

> Setelah kita melakukan perbaikan untuk invalid value data indicate currency, maka kita cek kembali data tersebut.

```
df["indicate_currency"].unique()
#hasil perbaikan penulisan format mata uang
```

```
array([nan, 'INR', 'ARS', 'MYR', 'CHF', 'KWD', 'NOK', 'USD', 'BRL', 'SEK',
       'CAD', 'DKK', 'EUR', 'COP', 'TTD', 'MXN', 'CZK', 'GBP', 'BDT',
       'ZAR', 'PHP', 'PLN', 'AUD', 'TRY', 'CNY', 'ILS', 'IDR', 'TWD',
       'NZD', 'SGD', 'KRW', 'THB', 'LKR', 'SAR', 'NTD', 'HRK', 'NGN'],
      dtype=object)
```

> Berikut merupakan data value dari column currency.

```
df["currency"].unique()
```

```
array(['USD', 'GBP', 'CAD', 'EUR', 'AUD', 'Other', 'CHF', 'ZAR', 'SEK',
       'HKD', 'JPY'], dtype=object)
```

> Selanjutnya dalam mengkonversi data dari kolum salary untuk merubah menjadi data salary dalam currency "USD", maka disini kita menggunakan bantuan dari library easy exchange rates.

```
!pip install easy-exchange-rates
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: easy-exchange-rates in c:\users\caturwarga computer\appda
```

> easy-exchange-rates adalah library Python yang menyediakan antarmuka yang mudah digunakan untuk mengambil nilai tukar dari berbagai sumber, seperti Bank Sentral Eropa (ECB) atau Federal Reserve Bank of St. Louis (FRED). Ini memungkinkan pengguna untuk mengonversi mata uang dan mengambil nilai tukar historis untuk analisis dan visualisasi.

```
from easy_exchange_rates import API
api = API()
# import library easy_exchange_rates untuk convert mata uang
```

> Selanjutnya kita membuat method yang nantinya akan diapply untuk konversi mata uang. Method pertama adalah convert_usd untuk mengkonversi salary dilihat dari data column indicate_currency. Sedangkan method kedua adalah sallary_in_usd untuk mengkonversi salary dilihat dari data column currency.

```
def convert_usd (row):
  result = api.get_exchange_rates(
      base_currency=row['indicate_currency'],
      start_date="2023-03-17",
      end_date="2023-03-17",
      targets=["USD"]
  )
  temp = api.to_dataframe(result).reset_index(drop=True)
  temp_x = temp['USD'].values.tolist()[0]
  return (temp_x * row["salary"])
```

```
def sallary_in_usd (row):
  result = api.get_exchange_rates(
      base_currency=row['currency'],
      start_date="2023-04-16",
      end_date="2023-04-16",
      targets=["USD"]
  )
  temp = api.to_dataframe(result).reset_index(drop=True)
  temp_x = temp['USD'].values.tolist()[0]
  return (temp_x * row["salary"])
```

> Pertama kita menginisialisasi data untuk column baru yaitu "USD Salary" dengan nilai value sama dengan value data di column "salary". Ini untuk data dimana currency == "USD".

> Tahap kedua adalah kita mengkonversi data salary dimana currency == "Other" dan indicate_currency != "USD" dengan mengaply method convert_usd

```
df["USD Salary"] = df['salary']
df.loc[(df["currency"] == 'Other') & (df["currency"] != 'USD') & (df["indicate_currency"] !=
```

> Tahap ketiga adalah kita mengkonversi data salary dimana currency != "Other" dan indicate_currency != "USD" dengan mengaply method sallary_in_usd

```
# 2. convert mata uang dengan value currency bukan other dan bukan USD dan value USD salary m
df.loc[(df["currency"] != 'Other') & (df["currency"] != 'USD'), 'USD Salary'] = df[(df["curre
# x = df[(df[  "currency"] != 'Other') & (df["currency"] != 'USD') & (df['USD Salary'].isna()
# len(x)
```

> Karena tahap konversi salary telah selesai, selanjutnya kita akan melakukan pengecekan apakah terdapat data USD Salary yang bernilai ("Nan").

```
x = df[(df["currency"] != 'Other') & (df["currency"] != 'USD') & (df['USD Salary'].isna() ==
len(x)
```

```
0
```

```
#3 isi value yang masih kosong pada kolom USD salary dengan data salary , karena sudah bertip
# df["USD Salary"] = df['salary'].fillna(df["USD Salary"])
df.isna().sum()
# df.sort_values("USD Salary" , ascending= False).head(5)
```

```
time                            0
age                             0
industry                        0
job_title                       0
need_additional_context     20653
salary                          0
monetary_compentation        7232
currency                        0
indicate_currency           27700
income_additional_contex    24842
country                         0
US_state                     4966
city                           75
work_experience_overall         0
work_experince_field            0
hights_education_complate     211
gender                        166
your_race                     168
USD Salary                      0
dtype: int64
```

> Dikarenakan proses konversi salary memakan waktu lama, oleh karena itu kita perlu menyimpan data yang sekarang ke dalam excel untuk sebagai backupan.

```
df.to_excel("Manager Salary Survey 2021.xlsx", index=False)
```

```
df.sort_values("USD Salary" , ascending= False).head(10)
#gaji tertinggi yaitu industri Utilities & Telecommunications dengan besar gaji 102000000.0
```

	time	age	industry	job_title	need_additional_context	salary
3605	2021-04-27 12:11:16.839	25-34	Utilities & Telecommunications	Operations Manager	NaN	10200000
26466	2021-06-14 04:19:17.179	55-64	Sales	Inside sales manager	NaN	500004
2124	2021-04-27 11:38:28.478	55-64	Art & Design	Owner and CEO	NaN	300000
15509	2021-04-28 17:09:29.493	35-44	Computing or Tech	Product Manager	NaN	211153
5755	2021-04-27 13:22:34.124	25-34	Health care	Attending Physician (general internal medicine)	NaN	190000
6783	2021-04-27	25-	Computing or Tech	Principal Software	NaN	165000

CLEANSING GENDER

9238	2021-04-27	35-	Public	Policy	NaN	133478

> Tahap selanjutnya adalah cleansing data column "Gender".

26558				Partner	sole proprietor LLC	130000

```
datas = pd.read_excel("Manager Salary Survey 2021.xlsx")
```

17708	21:11:38.595	34	Consulting	Consultant	Strategy field	126000

> Terdapat 5 jenis value data gender, untuk data kecuali woman dan man disini akan diganti dengan "?" karena sama-sama bukan termasuk jenis gender woman dan man.

```
datas['gender'].value_counts()
```

```
Woman                            21258
Man                               5417
Non-binary                         742
Other or prefer not to answer      290
Prefer not to answer                 1
Name: gender, dtype: int64
```

```
datas['gender'] = datas['gender'].replace('Non-binary', '?')
datas['gender'] = datas['gender'].replace('Other or prefer not to answer', '?')
```

```
datas['gender'] = datas['gender'].replace('Prefer not to answer', '?')
datas['gender'].value_counts()
```

```
Woman    21258
Man       5417
?         1033
Name: gender, dtype: int64
```

```
datas.to_excel("Manager Salary Survey 2021-FINAL.xlsx", index=False)
```

CLEANSING EDUCATION

```
datas['hights_education_complate'].value_counts()
```

```
College degree                       13432
Master's degree                       8817
Some college                          2043
PhD                                   1421
Professional degree (MD, JD, etc.)    1321
High School                            629
Name: hights_education_complate, dtype: int64
```

CLEANSING RACE

> Lalu, tahap selanjutnya adalah cleansing data column "race"

```
import re
```

```
datas = pd.read_excel("Manager Salary Survey 2021-FINAL.xlsx")
```

> Disini dilihat bahwa banyak tipe data dari value column "your_race".

```
datas['your_race'].unique()
```

```
array(['White', 'Hispanic, Latino, or Spanish origin, White',
       'Asian or Asian American, White', 'Asian or Asian American',
       'Another option not listed here or prefer not to answer',
       'Hispanic, Latino, or Spanish origin',
       'Middle Eastern or Northern African',
       'Hispanic, Latino, or Spanish origin, Middle Eastern or Northern African,
White',
```

```
       'Black or African American', 'Black or African American, White',
       nan,
       'Black or African American, Hispanic, Latino, or Spanish origin, White',
       'Native American or Alaska Native',
       'Native American or Alaska Native, White',
       'Hispanic, Latino, or Spanish origin, Another option not listed here or
prefer not to answer',
       'Black or African American, Middle Eastern or Northern African, Native
American or Alaska Native, White',
       'White, Another option not listed here or prefer not to answer',
       'Black or African American, Native American or Alaska Native, White',
       'Asian or Asian American, Another option not listed here or prefer not to
answer',
       'Middle Eastern or Northern African, White',
       'Asian or Asian American, Black or African American, White',
       'Black or African American, Hispanic, Latino, or Spanish origin',
       'Asian or Asian American, Black or African American',
       'Asian or Asian American, Hispanic, Latino, or Spanish origin, White',
       'Native American or Alaska Native, White, Another option not listed here or
prefer not to answer',
       'Asian or Asian American, Hispanic, Latino, or Spanish origin',
       'Asian or Asian American, Native American or Alaska Native, White',
       'Hispanic, Latino, or Spanish origin, Native American or Alaska Native',
       'Black or African American, Middle Eastern or Northern African, White',
       'Black or African American, Hispanic, Latino, or Spanish origin, Native
American or Alaska Native, White',
       'Black or African American, Another option not listed here or prefer not to
answer',
       'Native American or Alaska Native, Another option not listed here or prefer
not to answer',
       'Asian or Asian American, White, Another option not listed here or prefer not
to answer',
       'Asian or Asian American, Middle Eastern or Northern African',
       'Asian or Asian American, Hispanic, Latino, or Spanish origin, Native
American or Alaska Native, White',
       'Hispanic, Latino, or Spanish origin, Middle Eastern or Northern African',
       'Hispanic, Latino, or Spanish origin, Native American or Alaska Native,
White',
       'Middle Eastern or Northern African, White, Another option not listed here or
prefer not to answer',
       'Hispanic, Latino, or Spanish origin, White, Another option not listed here
or prefer not to answer',
       'Asian or Asian American, Black or African American, Hispanic, Latino, or
Spanish origin',
       'Asian or Asian American, Black or African American, Native American or
Alaska Native, White',
       'Middle Eastern or Northern African, Native American or Alaska Native,
White',
       'Asian or Asian American, Middle Eastern or Northern African, White',
       'Black or African American, Middle Eastern or Northern African',
```

> Selanjutnya kita akan menggolongkan, bahwa data yang bukan bernilai "White", "Black or Afrivan American", dan "Asian or Asian American" akan diubah valuenya menjadi "?"

```
temp3 = datas[(datas["your_race"] != 'White') & (datas["your_race"] != 'Asian or Asian Americ
temp3["your_race"].to_list()
```

```
['Hispanic, Latino, or Spanish origin, White',
 'Hispanic, Latino, or Spanish origin, White',
 'Asian or Asian American, White',
 'Another option not listed here or prefer not to answer',
 'Hispanic, Latino, or Spanish origin, White',
 'Hispanic, Latino, or Spanish origin',
 'Another option not listed here or prefer not to answer',
 'Asian or Asian American, White',
 'Hispanic, Latino, or Spanish origin, White',
 'Asian or Asian American, White',
 'Hispanic, Latino, or Spanish origin, White',
 'Middle Eastern or Northern African',
 'Hispanic, Latino, or Spanish origin, White',
 'Hispanic, Latino, or Spanish origin, Middle Eastern or Northern African, White',
 'Hispanic, Latino, or Spanish origin',
 'Hispanic, Latino, or Spanish origin',
 'Black or African American, White',
 'Another option not listed here or prefer not to answer',
 'Hispanic, Latino, or Spanish origin',
 'Hispanic, Latino, or Spanish origin, White',
 'Hispanic, Latino, or Spanish origin',
 'Asian or Asian American, White',
 'Hispanic, Latino, or Spanish origin',
 'Hispanic, Latino, or Spanish origin',
 'Asian or Asian American, White',
 'Another option not listed here or prefer not to answer',
 'Another option not listed here or prefer not to answer',
 'Asian or Asian American, White',
 'Hispanic, Latino, or Spanish origin, White',
 'Hispanic, Latino, or Spanish origin',
 'Hispanic, Latino, or Spanish origin',
 'Hispanic, Latino, or Spanish origin',
 'Hispanic, Latino, or Spanish origin',
 'Asian or Asian American, White',
 'Another option not listed here or prefer not to answer',
 'Another option not listed here or prefer not to answer',
 nan,
 'Another option not listed here or prefer not to answer',
 'Hispanic, Latino, or Spanish origin',
 'Hispanic, Latino, or Spanish origin, White',
 nan,
 'Hispanic, Latino, or Spanish origin, White',
 'Black or African American, Hispanic, Latino, or Spanish origin, White',
 'Another option not listed here or prefer not to answer',
 nan,
 'Hispanic, Latino, or Spanish origin, White',
 'Another option not listed here or prefer not to answer',
 'Another option not listed here or prefer not to answer',
 'Another option not listed here or prefer not to answer',
 'Another option not listed here or prefer not to answer',
 'Hispanic, Latino, or Spanish origin',
```

```
        'Native American or Alaska Native',
        'Hispanic, Latino, or Spanish origin, White',
        'Native American or Alaska Native, White',
        'Hispanic, Latino, or Spanish origin, White',
        'Asian or Asian American, White',
        'Asian or Asian American, White',
        'Hispanic, Latino, or Spanish origin, White',
```

```
datas['your_race'] = datas['your_race'].replace(temp3["your_race"].to_list(), '?')
datas['your_race'].value_counts()
```

```
White                         23106
?                              2723
Asian or Asian American        1379
Black or African American       666
Name: your_race, dtype: int64
```

```
datas.to_excel("Manager Salary Survey 2021-FINAL.xlsx", index=False)
```

CLEANSING COUNTRY

> Tahap selanjutnya kita akan melakukan cleansing untuk value column "country" menggunakan library dataprep.

```
!pip install dataprep --user
```

```
Requirement already satisfied: dataprep in c:\users\caturwarga computer\appdata\roam
Requirement already satisfied: tqdm<5.0,>=4.48 in d:\anaconda3\lib\site-packages (fr
Requirement already satisfied: bokeh<3,>=2 in d:\anaconda3\lib\site-packages (from d
Requirement already satisfied: nltk<4.0.0,>=3.6.7 in d:\anaconda3\lib\site-packages
Requirement already satisfied: pandas<2.0,>=1.1 in d:\anaconda3\lib\site-packages (f
Requirement already satisfied: sqlalchemy==1.3.24 in c:\users\caturwarga computer\ap
Requirement already satisfied: rapidfuzz<3.0.0,>=2.1.2 in c:\users\caturwarga comput
Requirement already satisfied: flask_cors<4.0.0,>=3.0.10 in c:\users\caturwarga comp
Requirement already satisfied: python-crfsuite==0.9.8 in c:\users\caturwarga compute
Requirement already satisfied: pydantic<2.0,>=1.6 in c:\users\caturwarga computer\ap
Requirement already satisfied: jsonpath-ng<2.0,>=1.5 in c:\users\caturwarga computer
Requirement already satisfied: wordcloud<2.0,>=1.8 in c:\users\caturwarga computer\a
Requirement already satisfied: ipywidgets<8.0,>=7.5 in d:\anaconda3\lib\site-package
Requirement already satisfied: aiohttp<4.0,>=3.6 in d:\anaconda3\lib\site-packages (
Requirement already satisfied: pydot<2.0.0,>=1.4.2 in c:\users\caturwarga computer\a
Requirement already satisfied: metaphone<0.7,>=0.6 in c:\users\caturwarga computer\a
Requirement already satisfied: regex<2022.0.0,>=2021.8.3 in c:\users\caturwarga comp
Requirement already satisfied: varname<0.9.0,>=0.8.1 in c:\users\caturwarga computer
Requirement already satisfied: jinja2<3.1,>=3.0 in c:\users\caturwarga computer\appd
Requirement already satisfied: dask[array,dataframe,delayed]>=2022.3.0 in c:\users\c
Requirement already satisfied: python-stdnum<2.0,>=1.16 in c:\users\caturwarga compu
Requirement already satisfied: flask<3,>=2 in c:\users\caturwarga computer\appdata\r
```

```
Requirement already satisfied: scipy<2.0,>=1.8 in c:\users\caturwarga computer\appda
Requirement already satisfied: numpy<2.0,>=1.21 in d:\anaconda3\lib\site-packages (f
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in d:\anaconda3\lib\site-
Requirement already satisfied: frozenlist>=1.1.1 in d:\anaconda3\lib\site-packages (
Requirement already satisfied: yarl<2.0,>=1.0 in d:\anaconda3\lib\site-packages (fro
Requirement already satisfied: attrs>=17.3.0 in d:\anaconda3\lib\site-packages (from
Requirement already satisfied: aiosignal>=1.1.2 in d:\anaconda3\lib\site-packages (f
Requirement already satisfied: charset-normalizer<3.0,>=2.0 in d:\anaconda3\lib\site
Requirement already satisfied: multidict<7.0,>=4.5 in d:\anaconda3\lib\site-packages
Requirement already satisfied: typing-extensions>=3.6.5 in c:\users\caturwarga compu
Requirement already satisfied: packaging>=16.8 in d:\anaconda3\lib\site-packages (fr
Requirement already satisfied: PyYAML>=3.10 in d:\anaconda3\lib\site-packages (from
Requirement already satisfied: tornado>=5.1 in d:\anaconda3\lib\site-packages (from
Requirement already satisfied: pillow>=7.1.0 in d:\anaconda3\lib\site-packages (from
Requirement already satisfied: partd>=1.2.0 in d:\anaconda3\lib\site-packages (from
Requirement already satisfied: fsspec>=0.6.0 in d:\anaconda3\lib\site-packages (from
Requirement already satisfied: cloudpickle>=1.1.1 in d:\anaconda3\lib\site-packages
Requirement already satisfied: importlib-metadata>=4.13.0 in c:\users\caturwarga com
Requirement already satisfied: toolz>=0.8.2 in d:\anaconda3\lib\site-packages (from
Requirement already satisfied: click>=7.0 in d:\anaconda3\lib\site-packages (from da
Requirement already satisfied: colorama in d:\anaconda3\lib\site-packages (from clic
Requirement already satisfied: Werkzeug>=2.2.2 in c:\users\caturwarga computer\appda
Requirement already satisfied: itsdangerous>=2.0 in d:\anaconda3\lib\site-packages (
Requirement already satisfied: Six in c:\users\caturwarga computer\appdata\roaming\p
Requirement already satisfied: zipp>=0.5 in d:\anaconda3\lib\site-packages (from imp
Requirement already satisfied: widgetsnbextension~=3.5.0 in d:\anaconda3\lib\site-pa
Requirement already satisfied: ipython>=4.0.0 in d:\anaconda3\lib\site-packages (fro
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in d:\anaconda3\lib\site-pa
Requirement already satisfied: nbformat>=4.2.0 in d:\anaconda3\lib\site-packages (fr
Requirement already satisfied: ipykernel>=4.5.1 in d:\anaconda3\lib\site-packages (f
Requirement already satisfied: ipython-genutils~=0.2.0 in d:\anaconda3\lib\site-pack
Requirement already satisfied: traitlets>=4.3.1 in d:\anaconda3\lib\site-packages (f
Requirement already satisfied: debugpy<2.0,>=1.0.0 in d:\anaconda3\lib\site-packages
```

> Library dataprep.clean adalah sebuah Python package yang menyediakan berbagai macam fungsi untuk membersihkan dan mempersiapkan data sebelum dilakukan analisis atau pemodelan data. Fungsi-fungsi tersebut meliputi:

1. clean_header: membersihkan header kolom
2. clean_lat_long: membersihkan koordinat latitude dan longitude
3. clean_email: membersihkan kolom email
4. clean_phone: membersihkan kolom nomor telepon
5. clean_address: membersihkan kolom alamat
6. clean_country: membersihkan kolom negara

```
from dataprep.clean import clean_country
```

```
Requirement already satisfied: pywin32>=1.0 in d:\anaconda3\lib\site-packages (from jupy
Requirement already satisfied: fastjsonschema in d:\anaconda3\lib\site-packages (from nb
```

```
Requirement already satisfied: jsonschema>=2.6 in d:\anaconda3\lib\site-packages (from r
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in d:\anaco
Requirement already satisfied: joblib in d:\anaconda3\lib\site-packages (from nltk<4.0.0
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in d:\anaconda3\lib\site-package
Requirement already satisfied: pytz>=2020.1 in d:\anaconda3\lib\site-packages (from panc
Requirement already satisfied: locket in d:\anaconda3\lib\site-packages (from partd>=1.2
Requirement already satisfied: wcwidth in d:\anaconda3\lib\site-packages (from prompt-to
Requirement already satisfied: asttokens<3.0.0,>=2.0.0 in d:\anaconda3\lib\site-packages
Requirement already satisfied: executing<0.9.0,>=0.8.3 in d:\anaconda3\lib\site-packages
Requirement already satisfied: pure_eval<1.0.0 in d:\anaconda3\lib\site-packages (from s
Requirement already satisfied: notebook>=4.4.1 in d:\anaconda3\lib\site-packages (from w
Requirement already satisfied: terminado>=0.8.3 in d:\anaconda3\lib\site-packages (from
Requirement already satisfied: prometheus-client in d:\anaconda3\lib\site-packages (from
Requirement already satisfied: Send2Trash>=1.8.0 in d:\anaconda3\lib\site-packages (from
Requirement already satisfied: nbconvert in d:\anaconda3\lib\site-packages (from noteboo
Requirement already satisfied: argon2-cffi in d:\anaconda3\lib\site-packages (from noteb
Requirement already satisfied: pywinpty>=1.1.0 in d:\anaconda3\lib\site-packages (from t
Requirement already satisfied: matplotlib in d:\anaconda3\lib\site-packages (from wordcl
Requirement already satisfied: idna>=2.0 in d:\anaconda3\lib\site-packages (from yarl<2
Requirement already satisfied: argon2-cffi-bindings in d:\anaconda3\lib\site-packages (f
Requirement already satisfied: cffi>=1.0.1 in d:\anaconda3\lib\site-packages (from argon
Requirement already satisfied: pycparser in d:\anaconda3\lib\site-packages (from cffi>=1
Requirement already satisfied: fonttools>=4.22.0 in d:\anaconda3\lib\site-packages (from
Requirement already satisfied: kiwisolver>=1.0.1 in d:\anaconda3\lib\site-packages (from
Requirement already satisfied: cycler>=0.10 in d:\anaconda3\lib\site-packages (from matp
Requirement already satisfied: defusedxml in d:\anaconda3\lib\site-packages (from nbconv
Requirement already satisfied: pandocfilters>=1.4.1 in d:\anaconda3\lib\site-packages (f
Requirement already satisfied: beautifulsoup4 in d:\anaconda3\lib\site-packages (from nb
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in d:\anaconda3\lib\site-packages
Requirement already satisfied: mistune<2,>=0.8.1 in d:\anaconda3\lib\site-packages (from
Requirement already satisfied: testpath in d:\anaconda3\lib\site-packages (from nbconver
Requirement already satisfied: bleach in d:\anaconda3\lib\site-packages (from nbconvert-
Requirement already satisfied: entrypoints>=0.2.2 in d:\anaconda3\lib\site-packages (fro
Requirement already satisfied: jupyterlab-pygments in d:\anaconda3\lib\site-packages (fr
Requirement already satisfied: soupsieve>1.2 in d:\anaconda3\lib\site-packages (from bea
Requirement already satisfied: webencodings in d:\anaconda3\lib\site-packages (from blea
```

```
datas['country'].unique()
```

```
array(['United States', 'United Kingdom', 'US', 'USA', 'Canada',
       'United Kingdom ', 'usa', 'UK', 'Scotland ', 'U.S.',
       'United States ', 'The Netherlands', 'Australia ', 'Spain', 'us',
       'Usa', 'England', 'finland', 'United States of America', 'France',
       'United states', 'Scotland', 'USA ', 'United states ', 'Germany',
       'UK ', 'united states', 'Ireland', 'India', 'Australia', 'Uk',
       'United States of America ', 'U.S. ', 'canada', 'Canada ', 'U.S>',
       'ISA', 'Argentina', 'Great Britain ', 'US ', 'United State',
       'U.S.A', 'Denmark', 'U.S.A.', 'America', 'Netherlands',
       'netherlands', 'England ', 'united states of america', 'Ireland ',
       'Switzerland', 'Netherlands ', 'Bermuda', 'Us',
       'The United States', 'United State of America', 'Germany ',
       'Malaysia', 'Mexico ', 'United Stated', 'South Africa ', 'Belgium',
       'Northern Ireland', 'u.s.', 'South Africa', 'UNITED STATES',
       'united States', 'Sweden', 'Hong Kong', 'Kuwait', 'Norway',
```

```
        'Sri lanka', 'Contracts', 'USA-- Virgin Islands', 'United Statws',
        'England/UK', 'U.S',
        "We don't get raises, we get quarterly bonuses, but they periodically asses
    income in the area you work, so I got a raise because a 3rd party assessment showed
    I was paid too little for the area we were located",
        'Unites States ', 'Usa ', 'U.S.A. ', 'England, UK.', 'Greece',
        'Japan', 'U. S. ', 'Britain ', 'United Sates', 'Japan ', 'Austria',
        'Brazil', 'Canada, Ottawa, ontario', 'Global', 'Sweden ',
        'United States of American ', 'FRANCE', 'Uniited States',
        'United Kingdom (England)',
        'Worldwide (based in US but short term trips aroudn the world)',
        'CANADA ', 'Canadw', 'Hungary', 'Luxembourg',
        'United Sates of America', 'ireland',
        'United States (I work from home and my clients are all over the
    US/Canada/PR',
        'Colombia', 'CANADA', 'Unted States', 'germany', 'United Statesp',
        'United Stattes', 'United Statea', 'United Kingdom.', 'Mexico',
        'New Zealand', 'Trinidad and Tobago', 'Unites States',
        'United Statees', 'United kingdom', 'Cayman Islands',
        'UNited States', 'Can',
        'I am located in Canada but I work for a company in the US',
        'United kingdom ', 'Uniyed states', 'Uniyes States',
        'United States of Americas', 'U.A.', 'Czech republic', 'Czechia',
        'Latvia', 'Finland', 'U. S.', 'Puerto Rico', 'US of A', 'Rwanda',
        'United States of america ', 'United Arab Emirates ',
        'Bangladesh ', 'Spain ', 'U.K. ', 'Romania', 'U.SA',
        'United Kindom', 'United Status', 'New Zealand ',
        'Currently finance', ' U.S.', 'Serbia', 'Philippines', 'Russia ',
        'Poland', 'UXZ', 'czech republic', 'England, UK', 'Turkey',
        'Canda', 'Puerto Rico ', 'Canada and USA', 'Catalonia',
        '$2,175.84/year is deducted for benefits', 'uk', 'France ',
        'Italy (South)', 'Jersey, Channel islands', 'Uk ', 'China',
        'Virginia', 'Afghanistan', 'Israel', 'U.s.', 'Great Britain',
        'U.s.a.', 'USS', 'Denmark ', 'Uniteed States',
        'New Zealand Aotearoa', 'U.K.', 'Hartford',
        'Japan, US Gov position', 'Csnada', 'United Stares', 'New zealand',
        'Mainland China', 'I.S.', 'UK (Northern Ireland)',
        'UK for U.S. company', ' US', 'Unites states ', 'NZ', 'Us ',
        'Hong Kong ', 'Taiwan', 'Canad', 'Unite States', 'australia',
        'The US', 'united states ', 'The Netherlands ', 'Cambodia',
        'United states of America ', 'Vietnam', 'Remote', 'Singapore',
        'South Korea', 'Czech Republic ', 'Thailand', 'Bangladesh',
        'Lithuania', 'Eritrea', 'Indonesia', 'Singapore ',
```

> Perintah dibawah akan melakukan clean country dari dataframe datas di column "Country". Hasil dari cleaning data tersebut akan disimpan di column baru yang bernama "country_clean".

```
datas = clean_country(datas, "country")
```

```
C:\Users\CATURWARGA COMPUTER\AppData\Roaming\Python\Python39\site-packages\dask\datafram
  warnings.warn(
  0%|
| 0/8 [00:00<…
Country Cleaning Report:
        14443 values cleaned (51.82%)
```

```
datas.head(10)
```

	time	age	industry	job_title	need_additional_context	salary
0	2021-04-27 11:02:09.743	25-34	Education (Higher Education)	Research and Instruction Librarian	NaN	55000
1	2021-04-27 11:02:21.562	25-34	Computing or Tech	Change & Internal Communications Manager	NaN	54600
2	2021-04-27 11:02:38.125	25-34	Accounting, Banking & Finance	Marketing Specialist	NaN	34000
3	2021-04-27 11:02:40.643	25-34	Nonprofits	Program Manager	NaN	62000
4	2021-04-27 11:02:41.793	25-34	Accounting, Banking & Finance	Accounting Manager	NaN	60000
5	2021-04-27 11:02:45.571	25-34	Education (Higher Education)	Scholarly Publishing Librarian	NaN	62000
6	2021-04-27 11:02:50.507	25-34	Publishing	Publishing Assistant	NaN	33000
7	2021-04-27 11:02:59.927	25-34	Education (Primary/Secondary)	Librarian	High school, FT	50000
8	2021-04-27 11:03:01.045	45-54	Computing or Tech	Systems Analyst	Data developer/ETL Developer	112000
9	2021-04-27 11:03:01.699	35-44	Accounting, Banking & Finance	Senior Accountant	NaN	45000

```
datas['country_clean'].value_counts()
```

```
United States             22886
Canada                     1673
United Kingdom              655
Australia                   385
Germany                     194
                           ...
Somalia                       1
Slovakia                      1
```

```
Sierra Leone                  1
Bahamas                       1
Bosnia and Herzegovina        1
Name: country_clean, Length: 95, dtype: int64
```

```
datas.to_excel("Manager Salary Survey 2021-FINAL.xlsx", index=False)
```

CLEANSING WORK EXPERIENCE OVERALL

```
datas = pd.read_excel("Manager Salary Survey 2021-FINAL.xlsx")
```

```
datas['work_experience_overall'].value_counts()
```

```
11 - 20 years       9577
8 - 10 years        5360
5-7 years           4848
21 - 30 years       3610
2 - 4 years         2990
31 - 40 years        863
1 year or less       506
41 years or more     120
Name: work_experience_overall, dtype: int64
```

CLEANSING WORK EXPERINCE FIELD

```
datas['work_experince_field'].value_counts()
```

```
11 - 20 years       6505
5-7 years           6488
2 - 4 years         6207
8 - 10 years        4949
21 - 30 years       1862
1 year or less      1448
31 - 40 years        377
41 years or more      38
Name: work_experince_field, dtype: int64
```

EDA

Dikarenakan tahap cleansing telah selesai maka kita dapat memulai untuk masuk ke tahap EDA.

```
df = pd.read_excel("Manager Salary Survey 2021-FINAL.xlsx")
```

```
df.head(5)
```

	time	age	industry	job_title	need_additional_context	salary	monetar
0	2021-04-27 11:02:09.743	25-34	Education (Higher Education)	Research and Instruction Librarian	NaN	55000	
1	2021-04-27 11:02:21.562	25-34	Computing or Tech	Change & Internal Communications Manager	NaN	54600	
2	2021-04-27 11:02:38.125	25-34	Accounting, Banking & Finance	Marketing Specialist	NaN	34000	
3	2021-04-27 11:02:40.643	25-34	Nonprofits	Program Manager	NaN	62000	
4	2021-04-27 11:02:41.793	25-34	Accounting, Banking & Finance	Accounting Manager	NaN	60000	

> Disini kita melakukan perubahan tipe data di column "USD Salary" yang awalnya tipe data float menjadi integer.

```
print(df["USD Salary"].dtypes)
df["USD Salary"] = df["USD Salary"].astype(int)
print(df["USD Salary"].dtypes)
```

```
float64
int32
```

Selanjutnya kita mengecek informasi dari dataframe.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27874 entries, 0 to 27873
Data columns (total 20 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   time                      27874 non-null  datetime64[ns]
 1   age                       27874 non-null  object
```

```
 2   industry                  27874 non-null  object
 3   job_title                 27874 non-null  object
 4   need_additional_context   7221 non-null   object
 5   salary                    27874 non-null  int64
 6   monetary_compentation     20642 non-null  float64
 7   currency                  27874 non-null  object
 8   indicate_currency         174 non-null    object
 9   income_additional_contex  3032 non-null   object
 10  country                   27874 non-null  object
 11  US_state                  22908 non-null  object
 12  city                      27799 non-null  object
 13  work_experience_overall   27874 non-null  object
 14  work_experince_field      27874 non-null  object
 15  hights_education_complate 27663 non-null  object
 16  gender                    27708 non-null  object
 17  your_race                 27874 non-null  object
 18  USD Salary                27874 non-null  int32
 19  country_clean             26738 non-null  object
dtypes: datetime64[ns](1), float64(1), int32(1), int64(1), object(16)
memory usage: 4.1+ MB
```

SALARRY FIELD

> EDA pertama yang akan dilakukan adalah data salary. Disini data salary merupakan data independen/data target. Dapat dilihat bahwa gaji tertinggi terletak di industry "Utilities & Telecommunications" dengan gaji tertinggi sebesar $102000000.

```
datas.sort_values("USD Salary" , ascending= False)
```

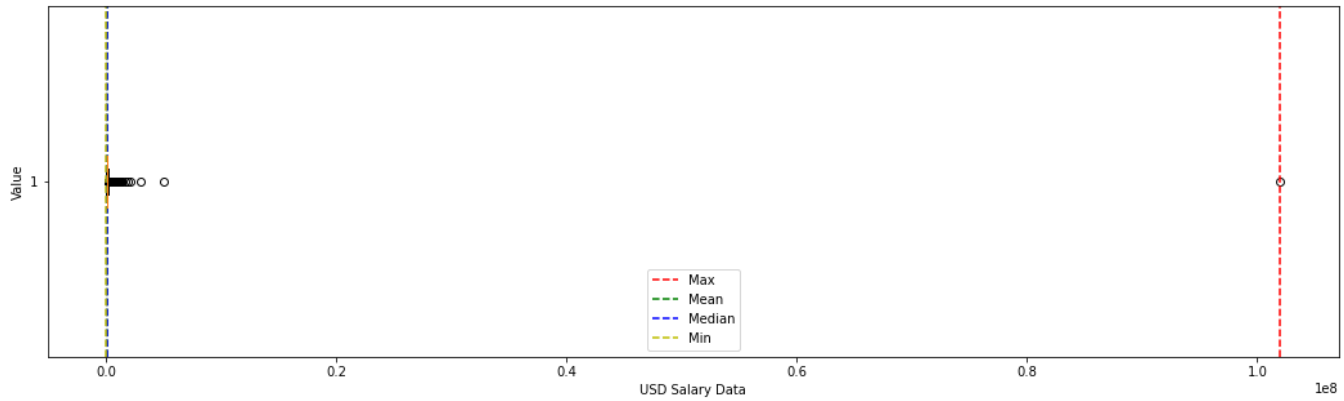
	time	age	industry	job_title	need_additional_context	sal
3597	2021-04-27 12:11:16.839	25-34	Utilities & Telecommunications	Operations Manager	NaN	102000
26405	2021-06-14 04:19:17.179	55-64	Sales	Inside sales manager	NaN	5000
2117	2021-04-27 11:38:28.478	55-64	Art & Design	Owner and CEO	NaN	3000
15481	2021-04-28 17:09:29.493	35-44	Computing or Tech	Product Manager	NaN	2111
5744	2021-04-27 13:22:34.124	25-34	Health care	Attending Physician (general internal medicine)	NaN	1900
...	...	...	...	...	...	

TAMPILAN GRAFIK BOX PLOT SEBELUM OUTLINER DIHILANGKAN

> Selanjutkan kita akan menampilkan grafik box plot data sebelum outliner dihilangkan. Output dari kode di bawah adalah grafik box plot dari data nilai dengan garis horizontal yang menunjukkan nilai maksimum (garis merah putus-putus), rata-rata (garis hijau putus-putus), dan media (garis biru putus-putus).

```
plt.figure(figsize=(18,5))
plt.boxplot(datas["USD Salary"], vert=False)

nilai_max = np.max(datas["USD Salary"])
nilai_mean = np.mean(datas["USD Salary"])
nilai_median = np.median(datas["USD Salary"])
nilai_min = np.min(datas["USD Salary"])

plt.axvline(nilai_max, color='r', linestyle='--', label='Max')
plt.axvline(nilai_mean, color='g', linestyle='--', label='Mean')
plt.axvline(nilai_median, color='b', linestyle='--', label='Median')
plt.axvline(nilai_min, color='y', linestyle='--', label='Min')

plt.legend()

plt.xlabel('USD Salary Data')
plt.ylabel('Value')
plt.show()
print("-----------------------------------------")
print("Max =", nilai_max)
```

```
print("Mean =", nilai_mean)
print("Median =", nilai_median)
print("Min =", nilai_min)
```




```
-------------------------------------------
Max = 102000000.0
Mean = 91061.75006534212
Median = 74000.0
Min = 0.0
```

> Dapat dilihat pada grafik di atas menampilkan informasi max, median, mean, dan min dari data sallary. Selanjutnya kita akan menghilangkan 1 outliner yaitu data salary yang bernilai $102000000.0.

```
# Disini kita menghilangkan 1 outliner
datas = datas[datas["USD Salary"] != 102000000.0]
#teradpat 1 outlier
len(df)
```

```
27874
```

> Dan ketika kita melakukan describe maka data max akan berubah menjadi data tertinggi nomor 2.

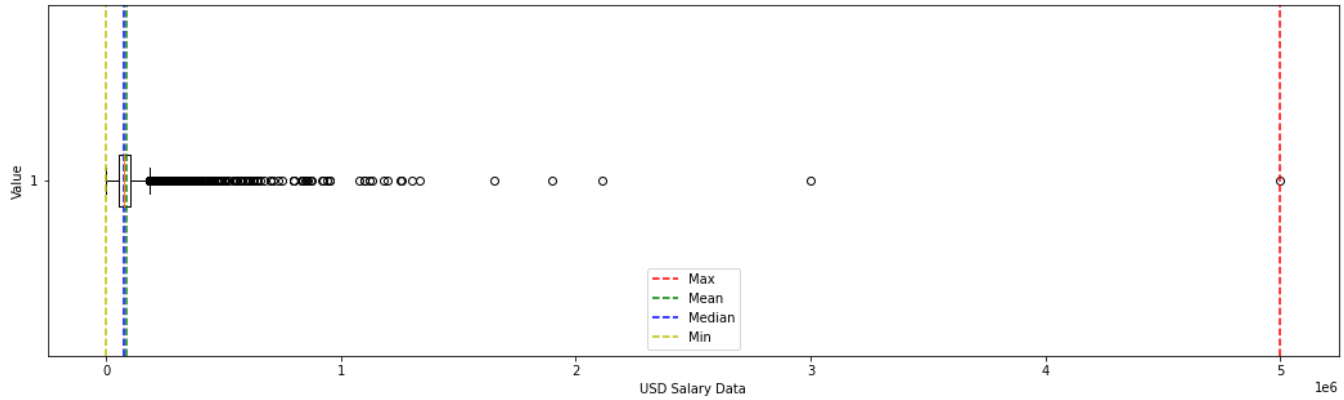
```
datas.describe()
```

	salary	monetary_compentation	USD Salary
count	2.787300e+04	2.064200e+04	2.787300e+04
mean	1.395242e+05	1.823604e+04	8.740556e+04
std	5.343529e+06	8.365536e+05	7.078717e+04
min	0.000000e+00	0.000000e+00	0.000000e+00
25%	5.400000e+04	0.000000e+00	5.300000e+04
50%	7.521900e+04	2.000000e+03	7.400000e+04

TAMPILAN GRAFIK BOX PLOT SETELAH OUTLINER DIHILANGKAN

> Selanjutnya kita akan melihat tampilan grafik box plot setelah outliner dihilangkan. Output dari kode di bawah adalah grafik box plot dari data nilai dengan garis horizontal yang menunjukkan nilai maksimum (garis merah putus-putus), rata-rata (garis hijau putus-putus), dan media (garis biru putus-putus).

```
plt.figure(figsize=(18,5))
plt.boxplot(datas["USD Salary"], vert=False)

nilai_max = np.max(datas["USD Salary"])
nilai_mean = np.mean(datas["USD Salary"])
nilai_median = np.median(datas["USD Salary"])
nilai_min = np.min(datas["USD Salary"])

plt.axvline(nilai_max, color='r', linestyle='--', label='Max')
plt.axvline(nilai_mean, color='g', linestyle='--', label='Mean')
plt.axvline(nilai_median, color='b', linestyle='--', label='Median')
plt.axvline(nilai_min, color='y', linestyle='--', label='Min')

plt.legend()

plt.xlabel('USD Salary Data')
plt.ylabel('Value')
plt.show()
print("-----------------------------------------")
print("Max =", nilai_max)
print("Mean =", nilai_mean)
print("Median =", nilai_median)
print("Min =", nilai_min)
```




```
-----------------------------------------
Max = 5000044.0
Mean = 87405.56170205389
Median = 74000.0
Min = 0.0
```

MENAMPILKAN INDUSTRI DENGAN GAJI TERTINGGI

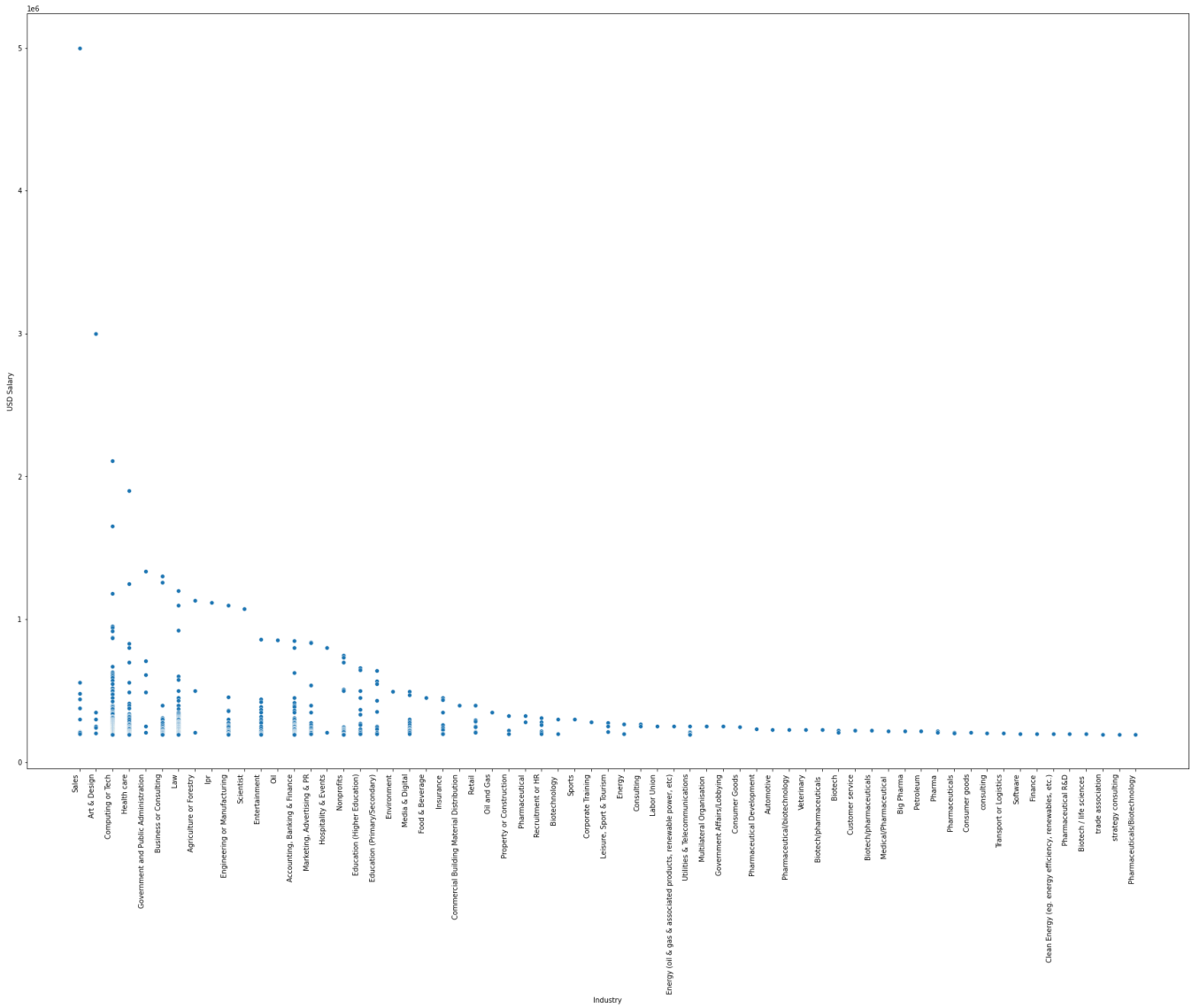
> Selanjutnya kita akan menampilkan visualisasi industri dengan gaji tertinggi. Disini kita akan membuat variabel baru yang bernama new_df untuk menyimpan data dimana "USD Salary" != 0 dan setelah itu diurutkan secara descending dengan mengambil 1000 data pertama.

```
new_df= datas[(datas["USD Salary"] != 0)].sort_values("USD Salary", ascending=False).head(100
z = new_df["job_title"].unique()
len(z)
```

```
621
```

> Selanjutnya kita akan melakukan visualisasi data dengan membuat sebuah scatter plot yang menampilkan hubungan antara USD Salary (sumbu y) dan industri pekerjaan (sumbu x) dari data frame "new_df".

```
plt.figure(figsize=(30,20))
sns.scatterplot(x=new_df["industry"], y=new_df["USD Salary"] , data=new_df)
plt.xlabel("Industry")
plt.ylabel("USD Salary")
plt.xticks(ha="right",rotation="90")
plt.show()
#1000 data industri gaji tertinggi
```


1e6
5
4
3
2
1
0
USD Salary
Sales
Art & Design
Computing or Tech
Health care
Government and Public Administration
Business or Consulting
Law
Agriculture or Forestry
Ipr
Engineering or Manufacturing
Scientist
Entertainment
Oil
Accounting, Banking & Finance
Marketing, Advertising & PR
Hospitality & Events
Nonprofits
Education (Higher Education)
Education (Primary/Secondary)
Environment
Media & Digital
Food & Beverage
Insurance
Commercial Building Material Distribution
Retail
Oil and Gas
Property or Construction
Pharmaceutical
Recruitment or HR
Biotechnology
Sports
Corporate Training
Leisure, Sport & Tourism
Energy
Consulting
Labor Union
Energy (oil & gas & associated products, renewable power, etc)
Utilities & Telecommunications
Multilateral Organisation
Government Affairs/Lobbying
Consumer Goods
Pharmaceutical Development
Automotive
Pharmaceutical/biotechnology
Veterinary
Biotech/pharmaceuticals
Biotech
Customer service
Biotech/pharmaceuticals
Medical/Pharmaceutical
Big Pharma
Petroleum
Pharma
Pharmaceuticals
Consumer goods
consulting
Transport or Logistics
Software
Finance
Clean Energy (eg. energy efficiency, renewables, etc.)
Pharmaceutical R&D
Biotech / life sciences
trade association
strategy consulting
Pharmaceuticals/Biotechnology
Industry

MENAMPILKAN INDUSTRI DENGAN GAJI TERENDAH

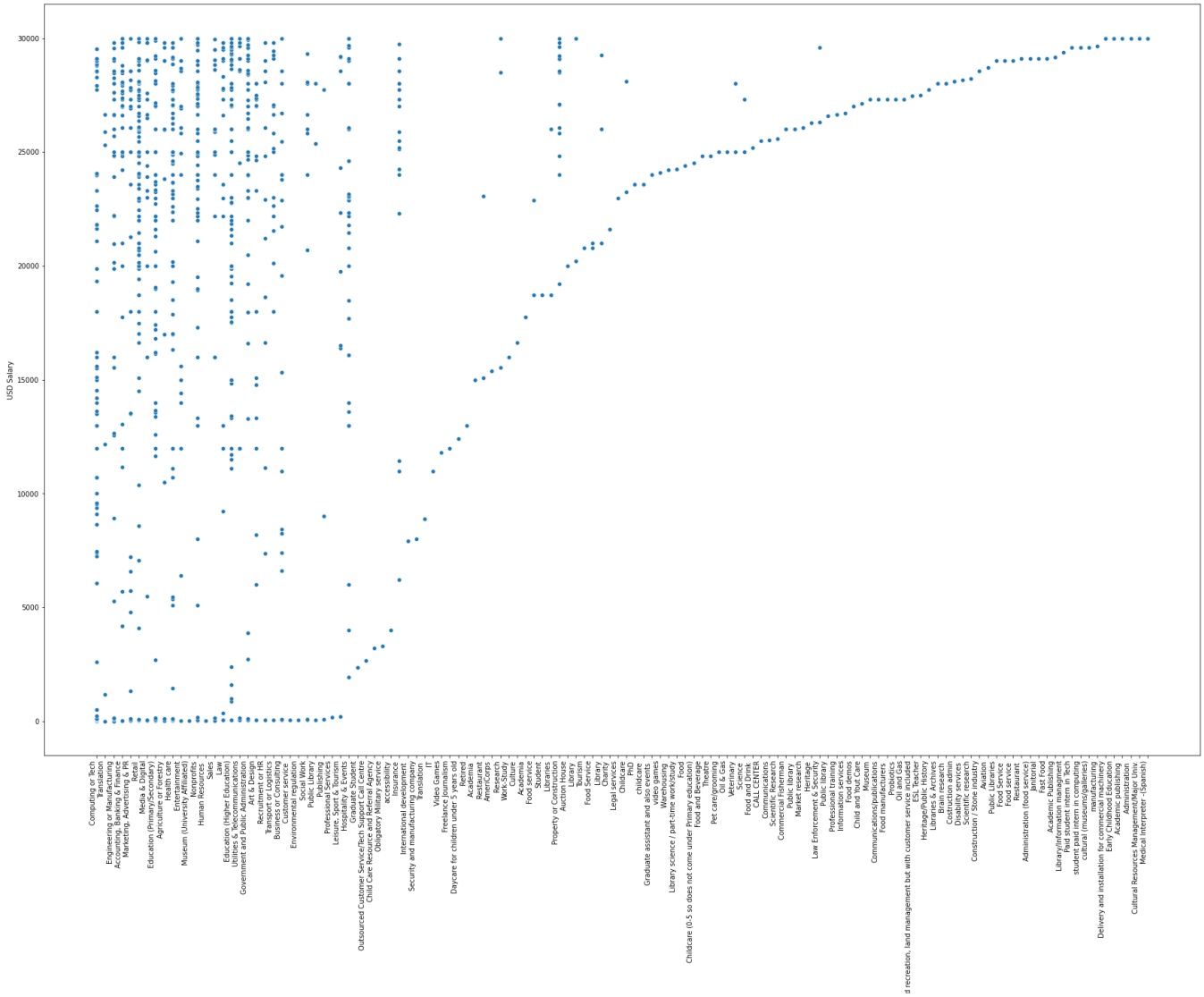
> Selanjutnya kita akan menampilkan visualisasi industri dengan gaji terendah. Disini kita akan membuat variabel baru yang bernama new_df untuk menyimpan data dimana "USD Salary" != 0 dan setelah itu diurutkan secara ascending dengan mengambil 1000 data pertama.

```
last_df= datas[(datas["USD Salary"] != 0)].sort_values("USD Salary", ascending=True).head(100
last_df.head(10)
```

	time	age	industry	job_title	need_additional_context	salary	mon
8981	2021-04-27 17:02:02.773	35-44	Computing or Tech	Software Development Lead	NaN	1	
8071	2021-04-27 15:34:47.009	25-34	Translation	Project Manager	NaN	72	
7529	2021-04-27 14:55:37.055	25-34	Engineering or Manufacturing	Administrative Assistant	Travel Coordinator	264	
15132	2021-04-28 16:29:36.993	25-34	Computing or Tech	Front End Engineer	NaN	102	
5868	2021-04-27 13:26:33.680	25-34	Computing or Tech	Programmer	NaN	18	
11889	2021-04-28 06:47:12.681	25-34	Accounting, Banking & Finance	Auditor	NaN	30	
15707	2021-04-28 17:16:33.084	25-34	Marketing, Advertising & PR	Media executive	Manage paid digital campaigns for clients at a...	28	
4072	2021-04-27 12:24:27.470	45-54	Retail	Operations Manager	NaN	35	
26150	2021-05-17 11:05:08.697	18-24	Media & Digital	Marketing Coordinator	NaN	36	
2222	2021-04-27 11:40:26.479	25-34	Marketing, Advertising & PR	Project Manager / Account Manager	NaN	33	

> Selanjutnya kita akan melakukan visualisasi data dengan membuat sebuah scatter plot yang menampilkan hubungan antara gaji dalam USD (sumbu y) dan industri pekerjaan (sumbu x) dari data frame "new_df".

```
plt.figure(figsize=(30,20))
sns.scatterplot(x=last_df["industry"], y=last_df["USD Salary"] , data=last_df)
plt.xlabel("Industry")
plt.ylabel("USD Salary")
plt.xticks(ha="right",rotation="90")
plt.show()
```



WORK EXPERINCE OVERALL

> Disini kita akan mengambil data sample di industri "Computing or Tech" dan menyimpannya di variabel new_df.

```
new_df = datas[(datas["industry"]== "Computing or Tech")]
```

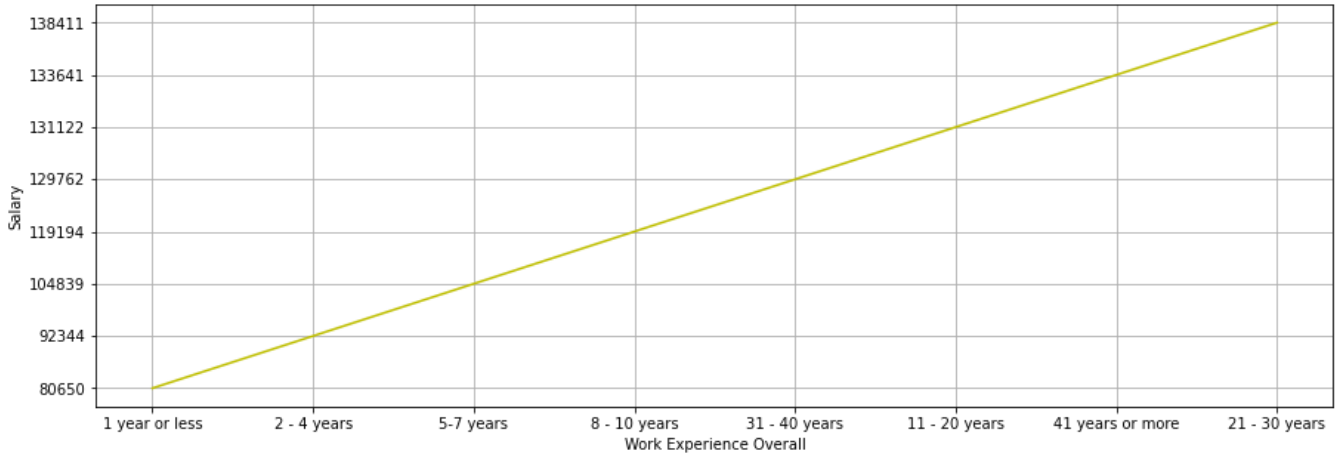
> Selanjutnya kita akan mengumbah type data column "work_experience_overall" yang bermula bertipe data object menjadi categories.

```
datas['work_experience_overall'] = pd.Categorical(datas['work_experience_overall'], categorie
```

> Selanjutnya kita akan membuat line plot yang menunjukkan nilai rata-rata dari tiap categori di column "work_experience_overall" terhadap data gaji ("USD Salary").

```
# INDUSTRY
b = new_df.groupby("work_experience_overall")["USD Salary"].mean().astype(int).sort_values()

plt.figure(figsize=(15,5))
plt.plot(b.index, b.values.astype(str), color='y' , label='mean')
plt.grid(10)
plt.xlabel("Work Experience Overall")
plt.ylabel("Salary")
plt.show()
```

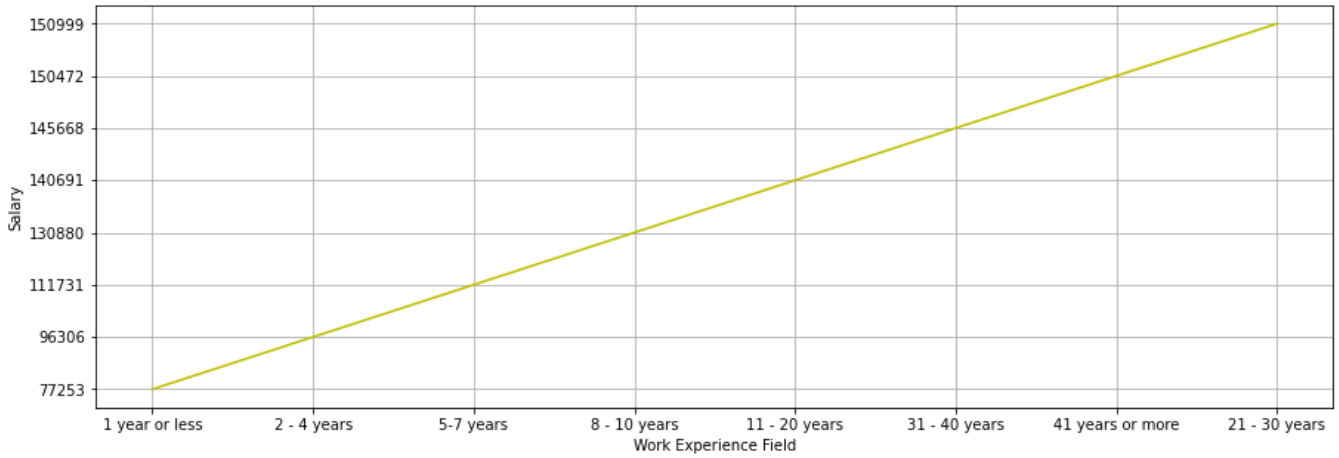
```
Using categorical units to plot a list of strings that are all parsable as floats or dat
Using categorical units to plot a list of strings that are all parsable as floats or dat
```



> Selanjutnya kita akan membuat line plot yang menunjukkan nilai rata-rata dari tiap categori di column "work_experience_field" terhadap data gaji ("USD Salary").

```
b = new_df.groupby("work_experince_field")["USD Salary"].mean().astype(int).sort_values()
plt.figure(figsize=(15,5))
plt.plot(b.index, b.values.astype(str), color='y' , label='mean')
plt.grid(10)
plt.xlabel("Work Experience Field")
plt.ylabel("Salary")
plt.show()
```

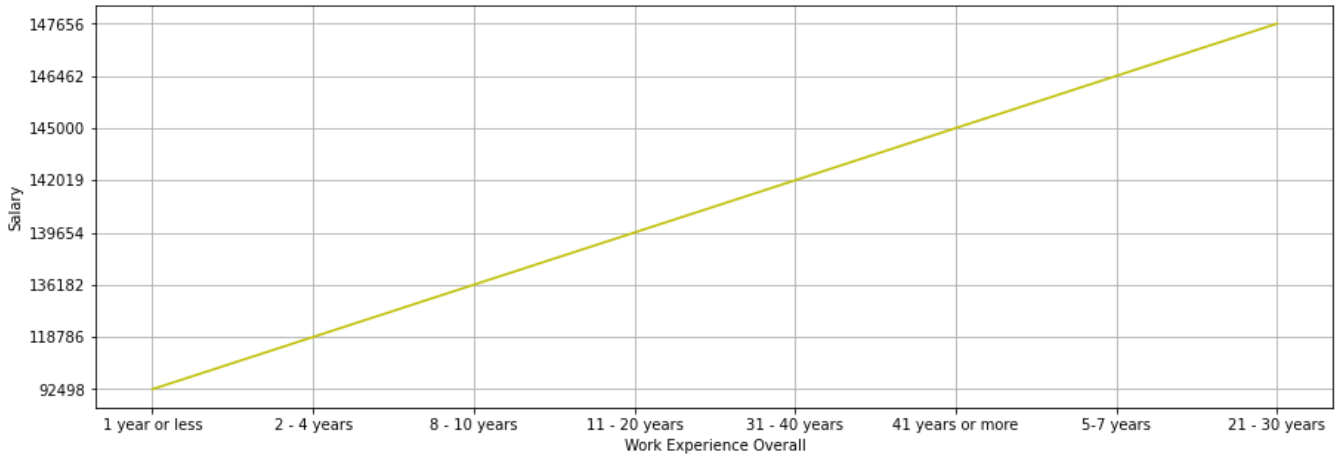
```
Using categorical units to plot a list of strings that are all parsable as floats or dat
Using categorical units to plot a list of strings that are all parsable as floats or dat
```



Disini mengambil data sample di job title "Software Engineer"

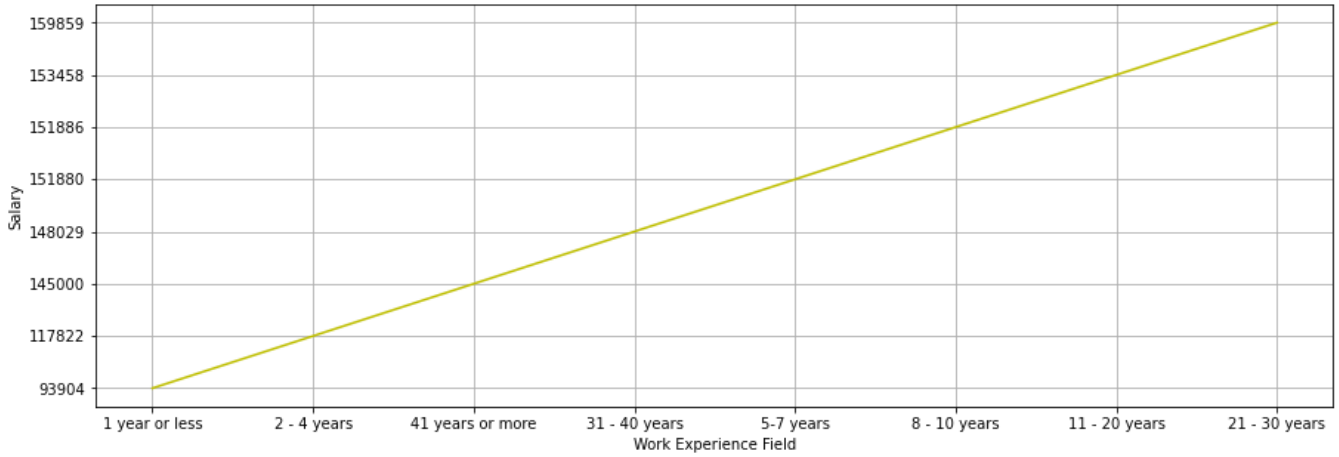
```
new_df = datas[(datas["job_title"]== "Software Engineer")]


b = new_df.groupby("work_experience_overall")["USD Salary"].mean().astype(int).sort_values()

plt.figure(figsize=(15,5))
plt.plot(b.index, b.values.astype(str), color='y' , label='mean')
plt.grid(10)
plt.xlabel("Work Experience Overall")
plt.ylabel("Salary")
plt.show()
```

```
Using categorical units to plot a list of strings that are all parsable as floats or dat
Using categorical units to plot a list of strings that are all parsable as floats or dat
```



```
b = new_df.groupby("work_experince_field")["USD Salary"].mean().astype(int).sort_values()
plt.figure(figsize=(15,5))
plt.plot(b.index, b.values.astype(str), color='y' , label='mean')
plt.grid(10)
plt.xlabel("Work Experience Field")
plt.ylabel("Salary")
plt.show()
```

```
Using categorical units to plot a list of strings that are all parsable as floats or dat
Using categorical units to plot a list of strings that are all parsable as floats or dat
```




```
datas['work_experience_overall'] = pd.Categorical(datas['work_experience_overall'], categorie
work_exp_ovl = datas.sort_values('work_experience_overall' , ascending = True)
```

```
work_exp_ovl['work_experience_overall'].unique()
```

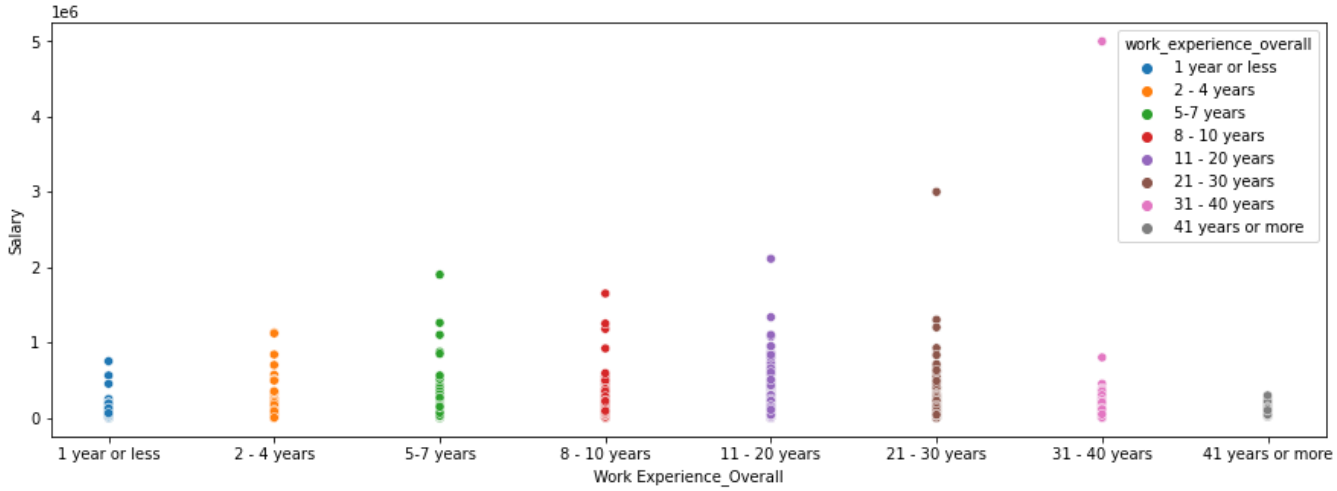
```
['1 year or less', '2 - 4 years', '5-7 years', '8 - 10 years', '11 - 20 years', '21 -
30 years', '31 - 40 years', '41 years or more']
Categories (8, object): ['1 year or less', '2 - 4 years', '5-7 years', '8 - 10 years',
'11 - 20 years', '21 - 30 years', '31 - 40 years', '41 years or more']
```

MELIHAT HUBUNGAN ANTARA PENGALAMAN KERJA SECARA KESELURUHAN DENGAN GAJI

> Kode di bawah akan membuat visualisasi scatterplot yang menunjukkan nilai data dari work_experience_overall (sumbu X) terhadap data USD Salary (sumbu Y).

```
plt.figure(figsize=(15,5))
sns.scatterplot(x= work_exp_ovl["work_experience_overall"], y =work_exp_ovl["USD Salary"] ,hu
plt.xlabel("Work Experience_Overall")
```

```
plt.ylabel("Salary")
plt.show()
```



> Perintah kode tersebut akan mengelompokkan data dalam DataFrame "work_exp_ovl" berdasarkan kolom "work_experience_overall" dan kemudian mengambil nilai gaji dalam USD (kolom "USD Salary") terendah,rata-rata, dan max dari setiap kelompok. Hasilnya akan disimpan ke dalam objek Series baru yang dinamakan "a", "b", dan "c" dengan tipe data integer.

```
a = work_exp_ovl.groupby("work_experience_overall")["USD Salary"].min().astype(int)
b = work_exp_ovl.groupby("work_experience_overall")["USD Salary"].mean().astype(int)
c = work_exp_ovl.groupby("work_experience_overall")["USD Salary"].max().astype(int)
print("Nilai min = ", a)
print("==========================================")
print("Nilai mean = ", b)
print("==========================================")
print("Nilai max = ", c)
```

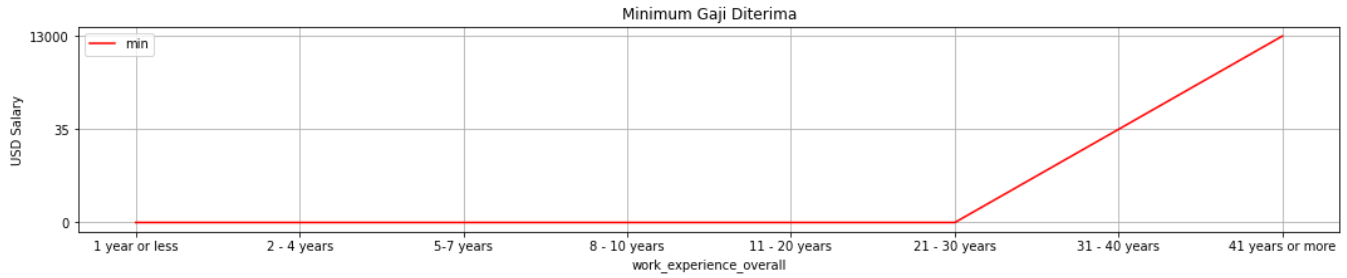
```
Nilai min =  work_experience_overall
1 year or less          0
2 - 4 years             0
5-7 years               0
8 - 10 years            0
11 - 20 years           0
21 - 30 years           0
31 - 40 years          35
41 years or more    13000
Name: USD Salary, dtype: int32
==========================================
Nilai mean =  work_experience_overall
```

```
1 year or less       62618
2 - 4 years          66777
5-7 years            76105
8 - 10 years         84592
11 - 20 years        95131
21 - 30 years       102784
31 - 40 years       103340
41 years or more     94197
Name: USD Salary, dtype: int32
========================================
Nilai max =  work_experience_overall
1 year or less       750000
2 - 4 years         1129776
5-7 years           1900000
8 - 10 years        1650000
11 - 20 years       2111538
21 - 30 years       3000000
31 - 40 years       5000044
41 years or more     295000
Name: USD Salary, dtype: int32
```

> Selanjutnya kita akan membuat lineplot untuk menampilkan data work_experience_overall (Sumbu X) terhadap USD Salary (Sumbu Y). Data yang divisualisasikan diambil dari variabel a yang sebelumnya merupakan data minimal gaji dari tiap-tiap kategori work_experience_overall.

```
plt.figure(figsize=(18,3))
plt.plot(a.index, a.values.astype(str) , color='r' , label='min')
plt.grid(10)
plt.xlabel("work_experience_overall")
plt.ylabel("USD Salary")
plt.title("Minimum Gaji Diterima")
plt.legend()
plt.show()
```

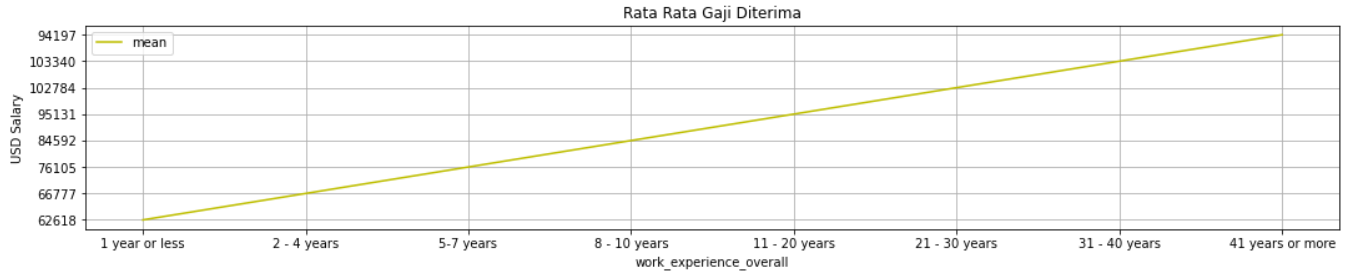
```
Using categorical units to plot a list of strings that are all parsable as floats or dat
Using categorical units to plot a list of strings that are all parsable as floats or dat
```



> Selanjutnya kita akan membuat lineplot untuk menampilkan data work_experience_overall (Sumbu X) terhadap USD Salary (Sumbu Y). Data yang divisualisasikan diambil dari variabel b yang sebelumnya merupakan data rata-rata gaji dari tiap-tiap kategori work_experience_overall.

```
plt.figure(figsize=(18,3))
#plt.plot(a.index, a.values.astype(str) , color='r' , label='min')
plt.plot(b.index, b.values.astype(str), color='y' , label='mean')
#plt.plot(c.index, c.values.astype(str), color='b' , label='max')
plt.grid(10)
plt.xlabel("work_experience_overall")
plt.ylabel("USD Salary")
plt.title("Rata Rata Gaji Diterima")
plt.legend()
plt.show()
```

```
Using categorical units to plot a list of strings that are all parsable as floats or dat
Using categorical units to plot a list of strings that are all parsable as floats or dat
```

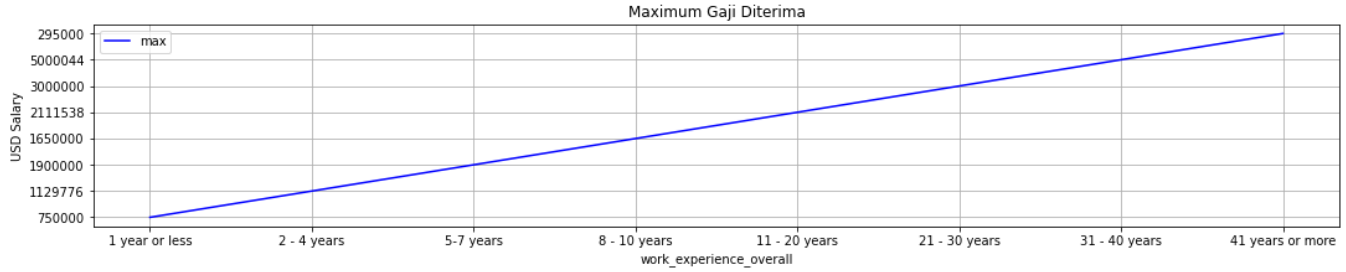


> Selanjutnya kita akan membuat lineplot untuk menampilkan data work_experience_overall (Sumbu X) terhadap USD Salary (Sumbu Y). Data yang divisualisasikan diambil dari variabel c yang sebelumnya merupakan data maksimal gaji dari tiap-tiap kategori work_experience_overall.

```
plt.figure(figsize=(18,3))
#plt.plot(a.index, a.values.astype(str) , color='r' , label='min')
#plt.plot(b.index, b.values.astype(str), color='y' , label='mean')
plt.plot(c.index, c.values.astype(str), color='b' , label='max')
plt.grid(10)
plt.xlabel("work_experience_overall")
plt.ylabel("USD Salary")
plt.title("Maximum Gaji Diterima")
```

```
plt.legend()
plt.show()
```

```
Using categorical units to plot a list of strings that are all parsable as floats or dat
Using categorical units to plot a list of strings that are all parsable as floats or dat
```



WORK EXPERINCE FIELD

```
datas['work_experince_field'].value_counts()
```

```
11 - 20 years       6505
5-7 years           6488
2 - 4 years         6207
8 - 10 years        4949
21 - 30 years       1862
1 year or less      1447
31 - 40 years        377
41 years or more      38
Name: work_experince_field, dtype: int64
```

```
datas['work_experince_field'] = pd.Categorical(datas['work_experince_field'], categories=['1
```

```
datas['work_experince_field'] = pd.Categorical(datas['work_experince_field'], categories=['1
work_exp_field = datas.sort_values('work_experince_field' , ascending = True)
```

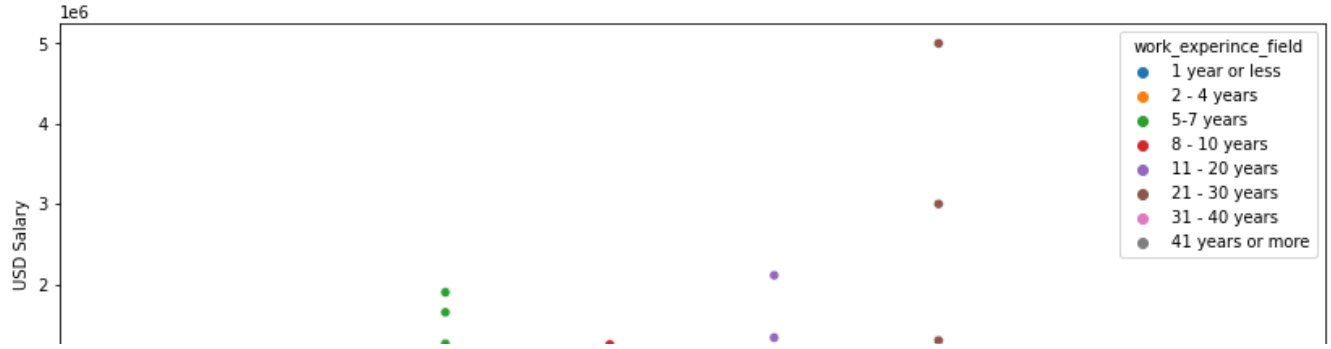
```
a = work_exp_field.groupby("work_experince_field")["USD Salary"].min().astype(int)
b = work_exp_field.groupby("work_experince_field")["USD Salary"].mean().astype(int)
c = work_exp_field.groupby("work_experince_field")["USD Salary"].max().astype(int)
a = a.sort_values()
b = b.sort_values()
c = c.sort_values()
print("Nilai min = ", a)
print("========================================")
print("Nilai mean = ", b)
```

```
print("=========================================")
print("Nilai max = ", c)
```

```
Nilai min =  work_experince_field
1 year or less          0
2 - 4 years             0
5-7 years               0
8 - 10 years            0
11 - 20 years           0
21 - 30 years           0
31 - 40 years          40
41 years or more    13000
Name: USD Salary, dtype: int32
=========================================
Nilai mean =  work_experince_field
1 year or less       59714
2 - 4 years          69339
5-7 years            81504
8 - 10 years         91810
41 years or more    101099
11 - 20 years       102677
31 - 40 years       109775
21 - 30 years       119840
Name: USD Salary, dtype: int32
=========================================
Nilai max =  work_experince_field
41 years or more     295000
31 - 40 years        450000
1 year or less       750000
2 - 4 years         1119703
8 - 10 years        1250000
5-7 years           1900000
11 - 20 years       2111538
21 - 30 years       5000044
Name: USD Salary, dtype: int32
```

MELIHAT HUBUNGAN ANTARA PENGALAMAN KERJA DI LAPANGAN DENGAN GAJI

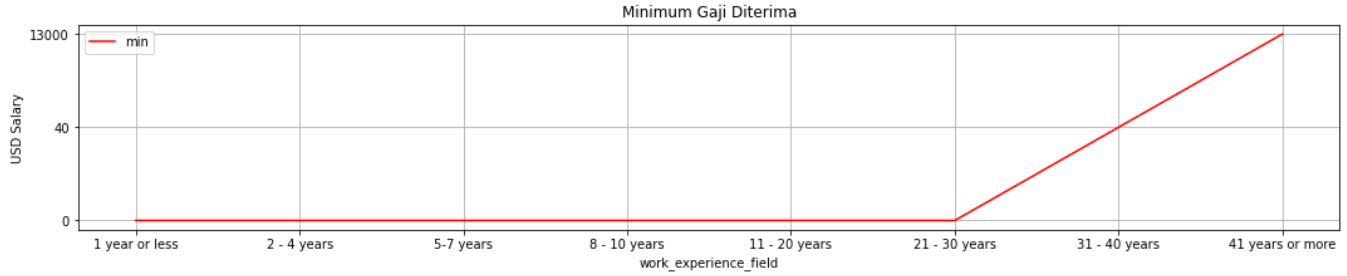
```
plt.figure(figsize=(14,5))
sns.scatterplot(x=work_exp_field["work_experince_field"], y=work_exp_field["USD Salary"] ,hue
plt.xlabel("Work Experience field")
plt.ylabel("USD Salary")
plt.show()
```



> Selanjutnya kita akan membuat lineplot untuk menampilkan data work_experience_field (Sumbu X) terhadap USD Salary (Sumbu Y). Data yang divisualisasikan diambil dari variabel a yang sebelumnya merupakan data minimal gaji dari tiap-tiap kategori work_experience_field.

```
plt.figure(figsize=(18,3))
plt.plot(a.index, a.values.astype(str) , color='r' , label='min')
#plt.plot(b.index, b.values.astype(str), color='y' , label='mean')
#plt.plot(c.index, c.values.astype(str), color='b' , label='max')
plt.grid(10)
plt.xlabel("work_experience_field")
plt.ylabel("USD Salary")
plt.title("Minimum Gaji Diterima")
plt.legend()
plt.show()
```

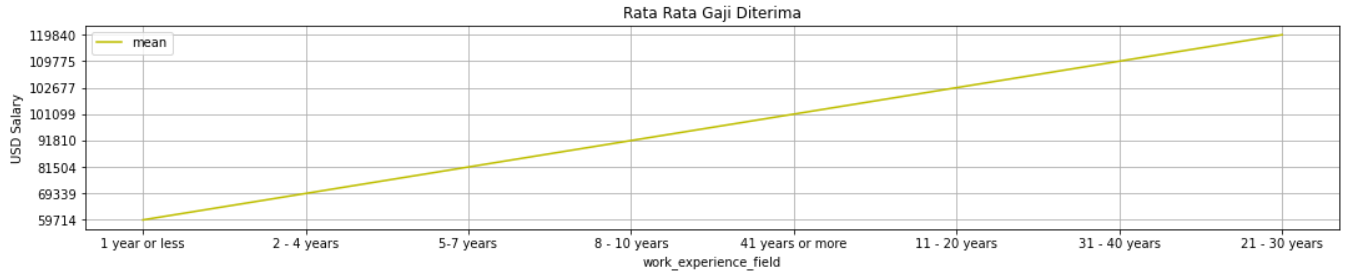
```
Using categorical units to plot a list of strings that are all parsable as floats or dat
Using categorical units to plot a list of strings that are all parsable as floats or dat
```



> Selanjutnya kita akan membuat lineplot untuk menampilkan data work_experience_field (Sumbu X) terhadap USD Salary (Sumbu Y). Data yang divisualisasikan diambil dari variabel b yang sebelumnya merupakan data rata-rata gaji dari tiap-tiap kategori work_experience_field.

```
plt.figure(figsize=(18,3))
#plt.plot(a.index, a.values.astype(str) , color='r' , label='min')
plt.plot(b.index, b.values.astype(str), color='y' , label='mean')
#plt.plot(c.index, c.values.astype(str), color='b' , label='max')
plt.grid(10)
plt.xlabel("work_experience_field")
plt.ylabel("USD Salary")
plt.title("Rata Rata Gaji Diterima")
plt.legend()
plt.show()
```

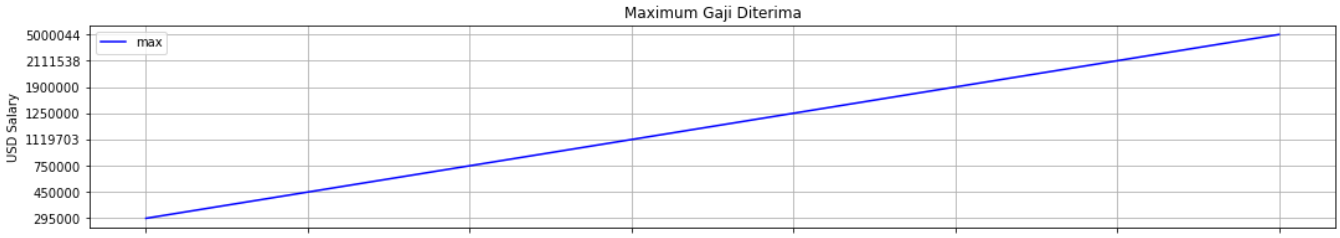
```
Using categorical units to plot a list of strings that are all parsable as floats or dat
Using categorical units to plot a list of strings that are all parsable as floats or dat
```



Selanjutnya kita akan membuat lineplot untuk menampilkan data work_experience_field (Sumbu X) terhadap USD Salary (Sumbu Y). Data yang divisualisasikan diambil dari variabel c yang sebelumnya merupakan data maksimum gaji dari tiap-tiap kategori work_experience_field.

```
plt.figure(figsize=(18,3))
#plt.plot(a.index, a.values.astype(str) , color='r' , label='min')
#plt.plot(b.index, b.values.astype(str), color='y' , label='mean')
plt.plot(c.index, c.values.astype(str), color='b' , label='max')
plt.grid(10)
plt.xlabel("work_experience_field")
plt.ylabel("USD Salary")
plt.title("Maximum Gaji Diterima")
plt.legend()
plt.show()
```

```
Using categorical units to plot a list of strings that are all parsable as floats or dat
Using categorical units to plot a list of strings that are all parsable as floats or dat
```



EDUCATION

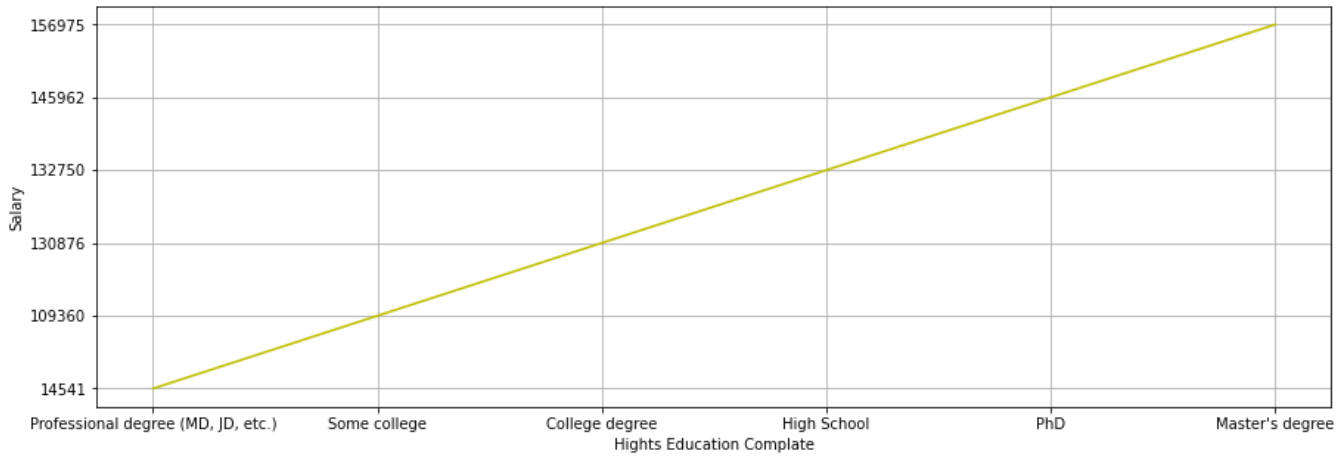
```
datas = datas[(datas["USD Salary"] != 102000000.0) & (datas["hights_education_complate"].isna
datas["hights_education_complate"].unique()
```

```
array(["Master's degree", 'College degree', 'PhD', 'Some college',
       'High School', 'Professional degree (MD, JD, etc.)'], dtype=object)
```

```
new_df = datas[(datas["job_title"]== "Software Engineer")]
```

```
b = new_df.groupby("hights_education_complate")["USD Salary"].mean().astype(int).sort_values(

plt.figure(figsize=(15,5))
plt.plot(b.index, b.values.astype(str), color='y' , label='mean')
plt.grid(10)
plt.xlabel("Hights Education Complate")
plt.ylabel("Salary")
plt.show()
```

```
Using categorical units to plot a list of strings that are all parsable as floats or dat
Using categorical units to plot a list of strings that are all parsable as floats or dat
```



```
datas['hights_education_complate'] = pd.Categorical(datas['hights_education_complate'], categ
education_complate = datas.sort_values('hights_education_complate' , ascending = True)
```

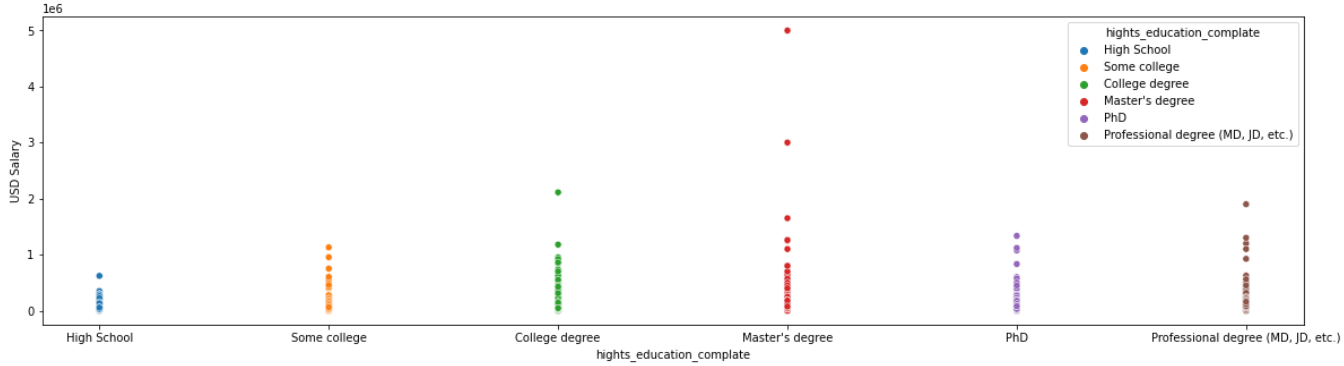
```
education_complate
```

	time	age	industry	job_title	need_additional_context	s
19914	2021-04-29 12:25:45.645	25-34	Computing or Tech	Communications Manager	NaN	1
18846	2021-04-29 04:21:57.600	35-44	Utilities & Telecommunications	Senior Network Engineer	NaN	
20915	2021-04-29 18:55:32.462	18-24	Transport or Logistics	Logistics technician	for tech equipment	
8774	2021-04-27 16:41:11.452	35-44	Utilities & Telecommunications	Industrial Mechanic	NaN	
845	2021-04-27 11:15:49.410	35-44	Insurance	HCE Analyst	HCE is HealthCare Economics	
...	...	...	...	...	...	
14709	2021-04-28 15:22:24.716	25-34	Education (Higher Education)	manager, educational administration	NaN	
13929	2021-04-28 13:50:16.847	35-44	Law	Commercial Counsel	In house counsel for global manufacturing/engi...	1
18724	2021-04-29 02:35:32.328	35-44	Government and Public Administration	Attorney Advisor	NaN	1
23680	2021-05-03 15:23:58.207	35-44	Government and Public Administration	Data and Analytics Lead	NaN	1
13211	2021-04-28 12:16:25.570	35-44	Pharmaceutical/Biotech	Program Manager	NaN	1

27662 rows × 20 columns

MELIHAT HUBUNGAN ANTARA JENJANG PENDIDIKAN TERHADAP GAJI

```
plt.figure(figsize=(20,5))
sns.scatterplot(x=education_complate["hights_education_complate"], y=education_complate["USD
plt.xlabel("hights_education_complate")
plt.ylabel("USD Salary")
plt.show()
```




```
a = education_complate.groupby("hights_education_complate")["USD Salary"].min().astype(int)
b = education_complate.groupby("hights_education_complate")["USD Salary"].mean().astype(int)
c = education_complate.groupby("hights_education_complate")["USD Salary"].max().astype(int)
a = a.sort_values()
b = b.sort_values()
c = c.sort_values()
print("Nilai min = ", a)
print("=========================================")
print("Nilai mean = ", b)
print("=========================================")
print("Nilai max = ", c)
```

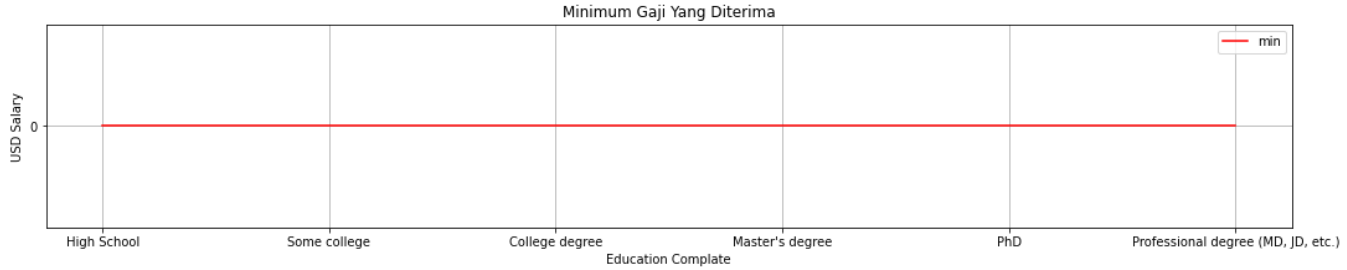
```
Nilai min =  hights_education_complate
High School                          0
Some college                         0
College degree                       0
Master's degree                      0
PhD                                  0
Professional degree (MD, JD, etc.)   0
Name: USD Salary, dtype: int32
=========================================
Nilai mean =  hights_education_complate
High School                          70403
Some college                         73229
College degree                       83032
Master's degree                      88943
PhD                                 104321
Professional degree (MD, JD, etc.)  133083
Name: USD Salary, dtype: int32
=========================================
Nilai max =  hights_education_complate
High School                          620756
Some college                        1129776
PhD                                 1334782
Professional degree (MD, JD, etc.)  1900000
```

```
College degree                    2111538
Master's degree                   5000044
Name: USD Salary, dtype: int32
```

> Selanjutnya kita akan membuat lineplot untuk menampilkan data education (Sumbu X) terhadap USD Salary (Sumbu Y). Data yang divisualisasikan diambil dari variabel a yang sebelumnya merupakan data minimum gaji dari tiap-tiap kategori education.

```
plt.figure(figsize=(18,3))
plt.plot(a.index, a.values.astype(str) , color='r' , label='min')
#plt.plot(b.index, b.values.astype(str), color='y' , label='mean')
#plt.plot(c.index, c.values.astype(str), color='b' , label='max')
plt.grid(10)
plt.xlabel("Education Complate")
plt.ylabel("USD Salary")
plt.title("Minimum Gaji Yang Diterima")
plt.legend()
plt.show()
```

```
Using categorical units to plot a list of strings that are all parsable as floats or dat
Using categorical units to plot a list of strings that are all parsable as floats or dat
```

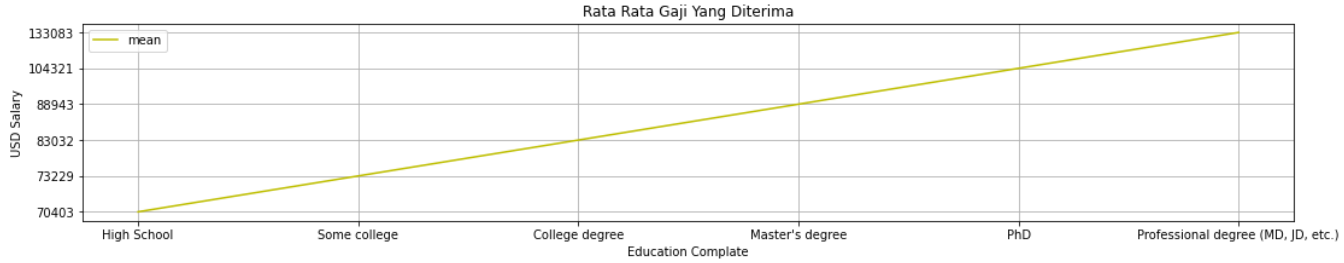


> Selanjutnya kita akan membuat lineplot untuk menampilkan data education (Sumbu X) terhadap USD Salary (Sumbu Y). Data yang divisualisasikan diambil dari variabel b yang sebelumnya merupakan data rata-rata gaji dari tiap-tiap kategori education.

```
plt.figure(figsize=(18,3))
#plt.plot(a.index, a.values.astype(str) , color='r' , label='min')
plt.plot(b.index, b.values.astype(str), color='y' , label='mean')
#plt.plot(c.index, c.values.astype(str), color='b' , label='max')
plt.grid(10)
plt.xlabel("Education Complate")
plt.ylabel("USD Salary")
plt.title("Rata Rata Gaji Yang Diterima")
```

```
plt.legend()
plt.show()
```

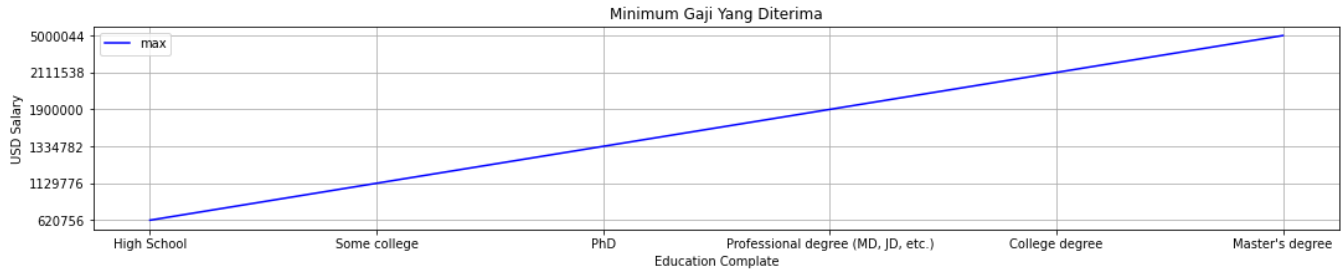
```
Using categorical units to plot a list of strings that are all parsable as floats or dat
Using categorical units to plot a list of strings that are all parsable as floats or dat
```



> Selanjutnya kita akan membuat lineplot untuk menampilkan data education (Sumbu X) terhadap USD Salary (Sumbu Y). Data yang divisualisasikan diambil dari variabel c yang sebelumnya merupakan data maksimum gaji dari tiap-tiap kategori education.

```
plt.figure(figsize=(18,3))
#plt.plot(a.index, a.values.astype(str) , color='r' , label='min')
#plt.plot(b.index, b.values.astype(str), color='y' , label='mean')
plt.plot(c.index, c.values.astype(str), color='b' , label='max')
plt.grid(10)
plt.xlabel("Education Complate")
plt.ylabel("USD Salary")
plt.title("Minimum Gaji Yang Diterima")
plt.legend()
plt.show()
```

```
Using categorical units to plot a list of strings that are all parsable as floats or dat
Using categorical units to plot a list of strings that are all parsable as floats or dat
```



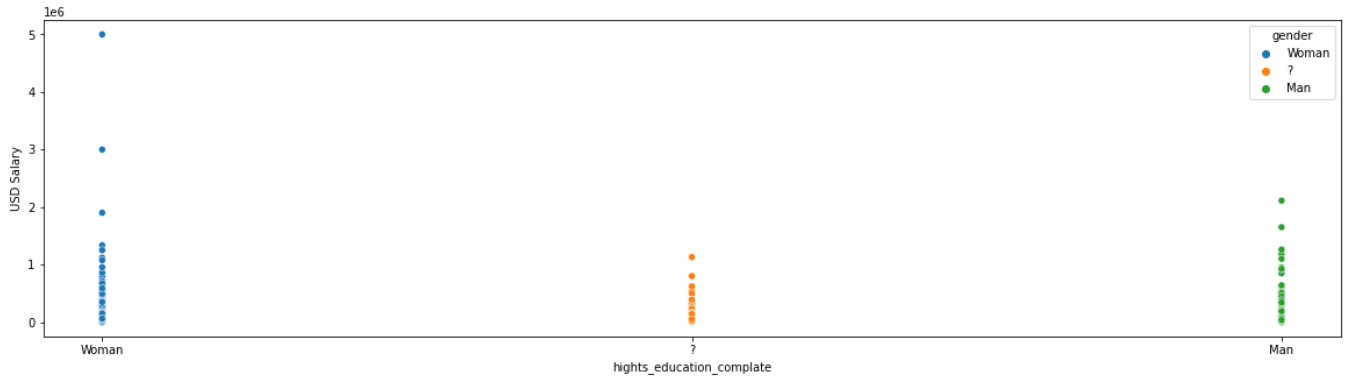
GENDER

```
datas = datas[(datas["gender"].isna() == False)]
datas['gender'].value_counts()
```

```
Woman    21130
Man       5376
?         1027
Name: gender, dtype: int64
```

MELIHAT HUBUNGAN GENDER DENGAN GAJI

```
plt.figure(figsize=(20,5))
sns.scatterplot(x=datas["gender"], y=datas["USD Salary"] ,hue=datas["gender"] , data=educatio
plt.xlabel("hights_education_complate")
plt.ylabel("USD Salary")
plt.show()
```




```
a = datas.groupby("gender")["USD Salary"].min().astype(int)
b = datas.groupby("gender")["USD Salary"].mean().astype(int)
c = datas.groupby("gender")["USD Salary"].max().astype(int)
a = a.sort_values()
b = b.sort_values()
c = c.sort_values()
print("Nilai min = ", a)
print("=========================================")
print("Nilai mean = ", b)
```

```
print("=========================================")
print("Nilai max = ", c)
```

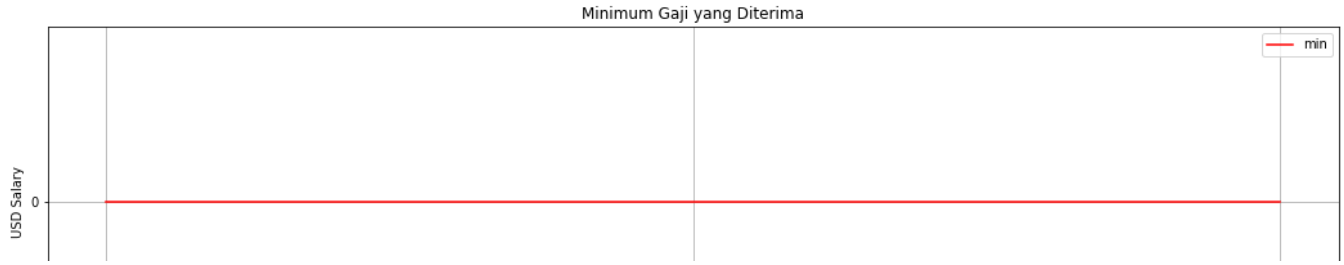
```
Nilai min =  gender
?        0
Man      0
Woman    0
Name: USD Salary, dtype: int32
=========================================
Nilai mean =  gender
?         76010
Woman     82798
Man      107182
Name: USD Salary, dtype: int32
=========================================
Nilai max =  gender
?        1129776
Man      2111538
Woman    5000044
Name: USD Salary, dtype: int32
```

MINIMUM GAJI BERDASARKAN GENDER

> Selanjutnya kita akan membuat lineplot untuk menampilkan data gender (Sumbu X) terhadap USD Salary (Sumbu Y). Data yang divisualisasikan diambil dari variabel a yang sebelumnya merupakan data minimum gaji dari tiap-tiap kategori gender.

```
#min gaji yang diterima
plt.figure(figsize=(18,5))
plt.plot(a.index, a.values.astype(str) , color='r' , label='min')
#plt.plot(b.index, b.values.astype(str), color='y' , label='mean')
#plt.plot(c.index, c.values.astype(str), color='b' , label='max')
plt.grid(10)
plt.xlabel("Gender")
plt.ylabel("USD Salary")
plt.title("Minimum Gaji yang Diterima")
plt.legend()
plt.show()
```

```
Using categorical units to plot a list of strings that are all parsable as floats or dat
Using categorical units to plot a list of strings that are all parsable as floats or dat
```

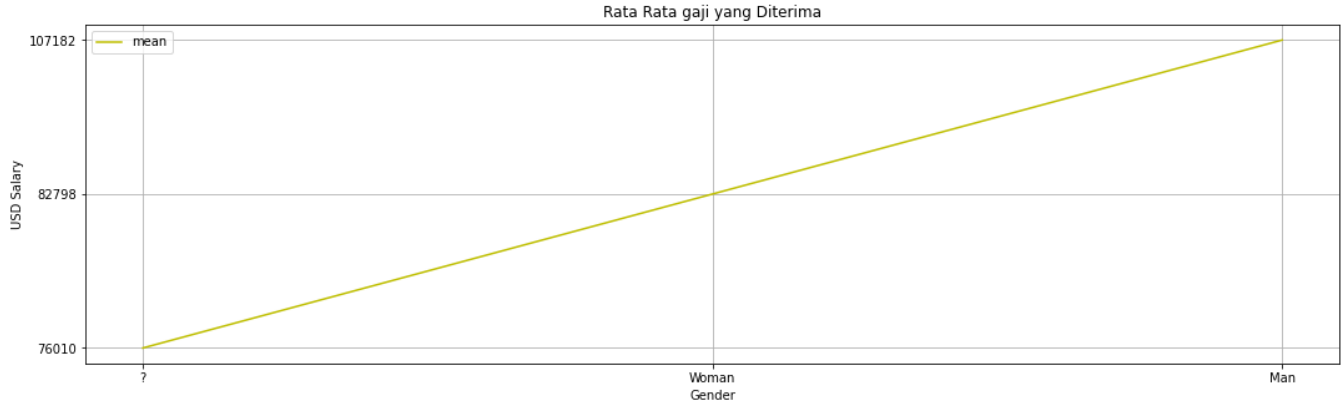


RATA RATA GAJI BERDASARKAN GENDER

> Selanjutnya kita akan membuat lineplot untuk menampilkan data gender (Sumbu X) terhadap USD Salary (Sumbu Y). Data yang divisualisasikan diambil dari variabel b yang sebelumnya merupakan data rata-rata gaji dari tiap-tiap kategori gender.

```
#rata rata gaji yang diterima
plt.figure(figsize=(18,5))
# plt.plot(a.index, a.values.astype(str) , color='r' , label='min')
plt.plot(b.index, b.values.astype(str), color='y' , label='mean')
#plt.plot(c.index, c.values.astype(str), color='b' , label='max')
plt.grid(10)
plt.xlabel("Gender")
plt.ylabel("USD Salary")
plt.title("Rata Rata gaji yang Diterima")
plt.legend()
plt.show()
```

```
Using categorical units to plot a list of strings that are all parsable as floats or dat
Using categorical units to plot a list of strings that are all parsable as floats or dat
```

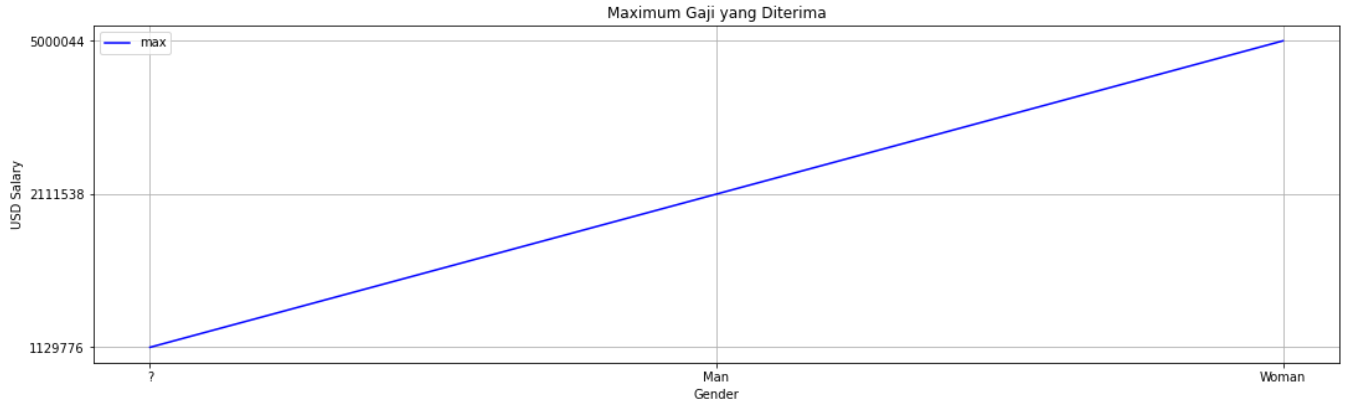


MAKSIMUM GAJI BERDASARKAN GENDER

> Selanjutnya kita akan membuat lineplot untuk menampilkan data gender (Sumbu X) terhadap USD Salary (Sumbu Y). Data yang divisualisasikan diambil dari variabel c yang sebelumnya merupakan data maksimum gaji dari tiap-tiap kategori gender.

```
#maximum gaji yang diterima
plt.figure(figsize=(18,5))
# plt.plot(a.index, a.values.astype(str) , color='r' , label='min')
# plt.plot(b.index, b.values.astype(str), color='y' , label='mean')
plt.plot(c.index, c.values.astype(str), color='b' , label='max')
plt.grid(10)
plt.xlabel("Gender")
plt.ylabel("USD Salary")
plt.title("Maximum Gaji yang Diterima")
plt.legend()
plt.show()
```

```
Using categorical units to plot a list of strings that are all parsable as floats or dat
Using categorical units to plot a list of strings that are all parsable as floats or dat
```



DISTRIBUSI SAMPLE

> Distribusi sample (atau distribusi sampel) adalah distribusi probabilitas dari sebuah sampel data yang diperoleh dari populasi tertentu. Distribusi sample menggambarkan frekuensi kemunculan nilai-nilai dalam sampel dan dapat membantu kita untuk memahami karakteristik populasi yang lebih besar. Distribusi sample dapat digunakan untuk menghitung nilai-nilai statistik seperti mean, variance, dan standar deviasi, yang kemudian dapat digunakan untuk membuat inferensi tentang populasi secara keseluruhan.

```
!pip install pyarrow --user
```

```
Requirement already satisfied: pyarrow in c:\users\caturwarga computer\appdata\roaming\python\python39\site-packages (11.0.0)
Requirement already satisfied: numpy>=1.16.6 in d:\anaconda3\lib\site-packages (from pyarrow) (1.21.5)
```

> Tahap pertama adalah melakukan import library yang akan digunakan.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import pyarrow
from scipy.stats import sem, t
import scipy.stats as stats
import statsmodels.api as sm
import statistics as stat
```

> Fungsi "pd.read_excel()" adalah fungsi panda yang membaca file Excel dan membuat objek DataFrame dari datanya. Argumen untuk fungsi tersebut adalah jalur file atau URL ke file Excel. Dalam hal ini, jalur file adalah "Manager Salary Survey 2021-FINAL (2).xlsx".

> Setelah kode dieksekusi, DataFrame "df" akan berisi data dari file Excel, yang kemudian dapat dimanipulasi dan dianalisis menggunakan panda dan pustaka Python lainnya.

```
df = pd.read_excel("Manager Salary Survey 2021-FINAL (2).xlsx")
```

> Selanjutnya kita dapat melihat 5 data teratas dari dataframe

```
df.head(5)
```

> Selanjutnya kita akan menghilangkan 1 outliner yaitu data salary yang bernilai $102000000.0.

```
df = df[df["USD Salary"] != 102000000.0]
```

> Selanjutnya kita membuat variabel baru untuk menyimpan data dimana dataset yang digunakan merupakan data dengan job_title yaitu "Software Engineer".

```
datas = df[(df["job_title"]== "Software Engineer")]
```

> Selanjutnya kita akan melihat tampilan grafik box plot setelah outliner dihilangkan. Output dari kode di bawah adalah grafik box plot dari data nilai dengan garis horizontal yang menunjukkan nilai maksimum (garis merah putus-putus), rata-rata (garis hijau putus-putus), dan media (garis biru putus-putus).

```
plt.figure(figsize=(18,5))
plt.boxplot(datas["USD Salary"], vert=False)

nilai_max = np.max(datas["USD Salary"])
nilai_mean = np.mean(datas["USD Salary"])
nilai_median = np.median(datas["USD Salary"])
nilai_min = np.min(datas["USD Salary"])

plt.axvline(nilai_max, color='r', linestyle='--', label='Max')
plt.axvline(nilai_mean, color='g', linestyle='--', label='Mean')
plt.axvline(nilai_median, color='b', linestyle='--', label='Median')
plt.axvline(nilai_min, color='y', linestyle='--', label='Min')

plt.legend()

plt.xlabel('USD Salary Data')
plt.ylabel('Value')
plt.show()
print("-------------------------------------------")
print("Max =", nilai_max)
print("Mean =", nilai_mean)
print("Median =", nilai_median)
print("Min =", nilai_min)
# Output dari kode di atas adalah grafik box plot pada data nilai dengan
# garis horizontal yang menunjukkan nilai maksimum (garis merah putus-putus), rata-rata (garis hijau putus-putus),
#dan median (garis biru putus-putus). Anda dapat memodifikasi tampilan grafik sesuai dengan kebutuhan Anda.
```

> Selanjutnya kita akan menghilangkan 1 outliner yaitu data salary yang bernilai $1100000.0.

```
# Disini kita menghilangkan 1 outliner
datas = datas[datas["USD Salary"] != 1100000.0]
#teradpat 1 outlier
len(df)
```

```
27873
```

> Dan ketika kita melakukan describe maka data max akan berubah menjadi data tertinggi nomor 2.

```
datas.describe()
```

NORMAL DISTRIBUTION (USD SALARY)

> Distribusi normal adalah distribusi probabilitas kontinu yang banyak digunakan dalam statistik, sains, teknik, dan keuangan. Ia juga dikenal sebagai distribusi Gaussian atau kurva lonceng, karena kurva berbentuk loncengnya yang simetris.

> Kode di bawah menghasilkan plot KDE (perkiraan kepadatan kernel) dari variabel "USD Salary" dari kumpulan data yang disebut "data" menggunakan pustaka seaborn dengan Python. Plot diwarnai hijau tua dan dibuat menggunakan parameter "kind" yang disetel ke "kde".

> Kode ini juga menyertakan anotasi yang menampilkan kemiringan dan kurtosis variabel "USD Salary" pada plot. Perulangan for mengulangi dua sumbu plot dan menambahkan nilai kemiringan dan kurtosis ke sudut kanan atas setiap sumbu menggunakan fungsi "ax.text". Nilai skewness dihitung menggunakan fungsi "skew" dari library pandas pada kolom "USD Salary" pada dataset, sedangkan nilai kurtosis dihitung menggunakan fungsi "kurt" dari library pandas pada kolom yang sama.

```
g = sns.displot(data=datas, x="USD Salary",kind ='kde',color='darkgreen')
for ax in g.axes.ravel():
    ax.text(x=0.97, y=0.97, transform=ax.transAxes, s="Skewness: %f" % df['USD Salary'].skew(),\
        fontsize=10, verticalalignment='top', horizontalalignment='right')
```

```
    ax.text(x=0.97, y=0.91, transform=ax.transAxes, s="Kurtosis: %f" % df['USD Salary'].kurt(),\
        fontsize=10, verticalalignment='top', horizontalalignment='right')
```

NORMAL DISTRIBUTION (MONETARY COMPETITION)

> Kode di bawah menghasilkan plot KDE (perkiraan kepadatan kernel) dari variabel "Monetary Competition" dari kumpulan data yang disebut "data" menggunakan pustaka seaborn dengan Python. Plot diwarnai hijau tua dan dibuat menggunakan parameter "kind" yang disetel ke "kde".

> Kode ini juga menyertakan anotasi yang menampilkan kemiringan dan kurtosis variabel "Monetary Competition" pada plot. Perulangan for mengulangi dua sumbu plot dan menambahkan nilai kemiringan dan kurtosis ke sudut kanan atas setiap sumbu menggunakan fungsi "ax.text". Nilai skewness dihitung menggunakan fungsi "skew" dari library pandas pada kolom "Monetary Competition" pada dataset, sedangkan nilai kurtosis dihitung menggunakan fungsi "kurt" dari library pandas pada kolom yang sama.

```
g = sns.displot(data=datas, x="monetary_compentation",kind ='kde',color='darkgreen')
for ax in g.axes.ravel():
    ax.text(x=0.97, y=0.97, transform=ax.transAxes, s="Skewness: %f" % df['monetary_compentation'].skew(),\
        fontsize=10, verticalalignment='top', horizontalalignment='right')
    ax.text(x=0.97, y=0.91, transform=ax.transAxes, s="Kurtosis: %f" % df['monetary_compentation'].kurt(),\
        fontsize=10, verticalalignment='top', horizontalalignment='right')
```

DISTRIBUTION SAMPLING (WORK EXPERINCE OVERALL)

> Sampling distribusi adalah proses menghasilkan sampel nilai acak dari distribusi probabilitas. Dalam statistik dan analisis data, pengambilan sampel dari suatu distribusi dapat digunakan untuk memperkirakan sifat-sifat distribusi, seperti mean, varians, dan kuantilnya.

> Salah satu cara untuk melakukan sampling distribusi adalah dengan menggunakan metode transformasi terbalik. Metode ini melibatkan transformasi variabel acak seragam pada interval [0,1] menjadi variabel acak dari distribusi yang diinginkan. Ini dilakukan dengan terlebih dahulu menemukan invers dari fungsi distribusi kumulatif (CDF) dari distribusi yang diinginkan, dan kemudian menerapkan fungsi invers ini ke variabel acak seragam. Nilai yang dihasilkan akan memiliki distribusi yang diinginkan.

```
datas['work_experience_overall'].value_counts()
```

```
2 - 4 years         72
5-7 years           70
11 - 20 years       58
8 - 10 years        55
1 year or less      13
21 - 30 years       12
31 - 40 years        3
41 years or more     1
Name: work_experience_overall, dtype: int64
```

```
data_wo = datas['work_experience_overall'].values.tolist()
```

> Kode menghasilkan plot batang menggunakan fungsi "barplot" seaborn. Sumbu x plot menunjukkan nilai unik dari variabel "Work Experience Overall", sedangkan sumbu y menunjukkan frekuensi setiap nilai unik.

> Baris "plt.figure(figsize=(20,10))" menetapkan ukuran plot menjadi 20 inci kali 10 inci.

> Garis "plt.xticks(ha='right',rotation='90')" memutar label sumbu x sebesar 90 derajat dan menyejajarkannya ke kanan.

> Baris "sns.barplot(x=pop_values,y=pop_frequencies,color='green')" membuat plot batang dengan "pop_values" pada sumbu x dan "pop_frequencies" pada sumbu y, berwarna hijau.

> Terakhir, kode menggunakan fungsi seaborn "despine" untuk menghapus duri atas dan kanan plot, dan menyetel label judul dan sumbu dengan fungsi "plt.title", "plt.xlabel", dan "plt.ylabel". .

> Secara keseluruhan, kode ini berguna untuk memvisualisasikan distribusi frekuensi variabel kategori seperti "Work Experience Overall" dalam kumpulan data.

```
pop_values, pop_frequencies = np.unique(data_wo, return_counts=True)

plt.figure(figsize=(20,10))
plt.xticks(ha="right",rotation="90")
sns.barplot(x=pop_values,y=pop_frequencies,color = 'green')

# Plot formatting
plt.title('Frequency chart of Population - Work Experience Overall distribution',fontsize='14')
sns.despine()
plt.xlabel('Work Experience Overall',fontsize='12')
plt.ylabel('Frequency',fontsize='12')
```

DISTRIBUTION SAMPLING (WORK EXPERIENCE FIELD)

> Kode menghasilkan plot batang menggunakan fungsi "barplot" seaborn. Sumbu x plot menunjukkan nilai unik dari variabel "Work Experience Field", sedangkan sumbu y menunjukkan frekuensi setiap nilai unik.

> Baris "plt.figure(figsize=(20,10))" menetapkan ukuran plot menjadi 20 inci kali 10 inci.

> Garis "plt.xticks(ha='right',rotation='90')" memutar label sumbu x sebesar 90 derajat dan menyejajarkannya ke kanan.

> Baris "sns.barplot(x=pop_values,y=pop_frequencies,color='green')" membuat plot batang dengan "pop_values" pada sumbu x dan "pop_frequencies" pada sumbu y, berwarna hijau.

> Terakhir, kode menggunakan fungsi seaborn "despine" untuk menghapus duri atas dan kanan plot, dan menyetel label judul dan sumbu dengan fungsi "plt.title", "plt.xlabel", dan "plt.ylabel". .

> Secara keseluruhan, kode ini berguna untuk memvisualisasikan distribusi frekuensi variabel kategori seperti "Work Experience Field" dalam kumpulan data.

```
pop_values, pop_frequencies = np.unique(datas['work_experince_field'].values.tolist(), return_counts=True)

plt.figure(figsize=(20,10))
plt.xticks(ha="right",rotation="90")
sns.barplot(x=pop_values,y=pop_frequencies,color = 'red')

# Plot formatting
plt.title('Frequency chart of Population - Work Experience Field distribution',fontsize='14')
sns.despine()
plt.xlabel('Work Experience Overall',fontsize='12')
plt.ylabel('Frequency',fontsize='12')
```

DISTRIBUTION SAMPLING (EDUCATION)

> Kode menghasilkan plot batang menggunakan fungsi "barplot" seaborn. Sumbu x plot menunjukkan nilai unik dari variabel "Education", sedangkan sumbu y menunjukkan frekuensi setiap nilai unik.

> Baris "plt.figure(figsize=(20,10))" menetapkan ukuran plot menjadi 20 inci kali 10 inci.

> Garis "plt.xticks(ha='right',rotation='90')" memutar label sumbu x sebesar 90 derajat dan menyejajarkannya ke kanan.

> Baris "sns.barplot(x=pop_values,y=pop_frequencies,color='green')" membuat plot batang dengan "pop_values" pada sumbu x dan "pop_frequencies" pada sumbu y, berwarna hijau.

> Terakhir, kode menggunakan fungsi seaborn "despine" untuk menghapus duri atas dan kanan plot, dan menyetel label judul dan sumbu dengan fungsi "plt.title", "plt.xlabel", dan "plt.ylabel". .

> Secara keseluruhan, kode ini berguna untuk memvisualisasikan distribusi frekuensi variabel kategori seperti "Education" dalam kumpulan data.

```
pop_values, pop_frequencies = np.unique(datas['hights_education_complate'].values.tolist(), return_counts=True)

plt.figure(figsize=(20,10))
plt.xticks(ha="right",rotation="90")
sns.barplot(x=pop_values,y=pop_frequencies,color = 'purple')

# Plot formatting
plt.title('Frequency chart of Population - Work Experience Overall distribution',fontsize='14')
sns.despine()
plt.xlabel('Work Experience Overall',fontsize='12')
plt.ylabel('Frequency',fontsize='12')
```

DISTRIBUTION SAMPLING (GENDER)

> Kode menghasilkan plot batang menggunakan fungsi "barplot" seaborn. Sumbu x plot menunjukkan nilai unik dari variabel "Gender", sedangkan sumbu y menunjukkan frekuensi setiap nilai unik.

> Baris "plt.figure(figsize=(20,10))" menetapkan ukuran plot menjadi 20 inci kali 10 inci.

> Garis "plt.xticks(ha='right',rotation='90')" memutar label sumbu x sebesar 90 derajat dan menyejajarkannya ke kanan.

> Baris "sns.barplot(x=pop_values,y=pop_frequencies,color='green')" membuat plot batang dengan "pop_values" pada sumbu x dan "pop_frequencies" pada sumbu y, berwarna hijau.

> Terakhir, kode menggunakan fungsi seaborn "despine" untuk menghapus duri atas dan kanan plot, dan menyetel label judul dan sumbu dengan fungsi "plt.title", "plt.xlabel", dan "plt.ylabel". .

> Secara keseluruhan, kode ini berguna untuk memvisualisasikan distribusi frekuensi variabel kategori seperti "Gender" dalam kumpulan data.

```
datas = datas[(datas["gender"].isna() == False)]


pop_values, pop_frequencies = np.unique(datas['gender'].values.tolist(), return_counts=True)

plt.figure(figsize=(20,10))
plt.xticks(ha="right",rotation="90")
sns.barplot(x=pop_values,y=pop_frequencies,color = 'Aqua')

# Plot formatting
plt.title('Frequency chart of Population - Work Experience Overall distribution',fontsize='14')
sns.despine()
plt.xlabel('Work Experience Overall',fontsize='12')
plt.ylabel('Frequency',fontsize='12')
```

DISTRIBUTION SAMPLING (COUNTRY)

```
datas['country_clean'].value_counts()
```

```
United States     230
Canada             11
United Kingdom      6
Germany             6
Australia           4
France              4
Ireland             2
Austria             2
India               2
Israel              2
Brazil              1
Switzerland         1
Sweden              1
Netherlands         1
Greece              1
Luxembourg          1
Name: country_clean, dtype: int64
```

```
from pandas.api.types import CategoricalDtype
# define the order of categories
cat_order = CategoricalDtype(categories=['United States', 'Canada', 'United Kingdom', 'Germany', 'Australia', 'France',
                                         'Ireland', 'Austria', 'India', 'Israel', 'Brazil', 'Switzerland', 'Sweden',
                                        'Netherlands', 'Greece', 'Luxembourg'], ordered=True)


# convert the column to categorical type with ordered=True
df['country_clean'] = df['country_clean'].astype(cat_order)


data_country = datas['country_clean'].values.tolist()
```

> Kode menghasilkan plot batang menggunakan fungsi "barplot" seaborn. Sumbu x plot menunjukkan nilai unik dari variabel "Country Clean", sedangkan sumbu y menunjukkan frekuensi setiap nilai unik.

> Baris "plt.figure(figsize=(20,10))" menetapkan ukuran plot menjadi 20 inci kali 10 inci.

> Garis "plt.xticks(ha='right',rotation='90')" memutar label sumbu x sebesar 90 derajat dan menyejajarkannya ke kanan.

> Baris "sns.barplot(x=pop_values,y=pop_frequencies,color='green')" membuat plot batang dengan "pop_values" pada sumbu x dan "pop_frequencies" pada sumbu y, berwarna hijau.

> Terakhir, kode menggunakan fungsi seaborn "despine" untuk menghapus duri atas dan kanan plot, dan menyetel label judul dan sumbu dengan fungsi "plt.title", "plt.xlabel", dan "plt.ylabel". .

> Secara keseluruhan, kode ini berguna untuk memvisualisasikan distribusi frekuensi variabel kategori seperti "Country Clean" dalam kumpulan data.

```
pop_values, pop_frequencies = np.unique(data_country, return_counts=True)

plt.figure(figsize=(20,10))
plt.xticks(ha="right",rotation="90")
sns.barplot(x=pop_values,y=pop_frequencies,color = 'pink')

# Plot formatting
plt.title('Frequency chart of Population - Country distribution',fontsize='14')
sns.despine()
plt.xlabel('Country',fontsize='12')
plt.ylabel('Frequency',fontsize='12')
```

DISTRIBUTION SAMPLING (AGE)

> Kode menghasilkan plot batang menggunakan fungsi "barplot" seaborn. Sumbu x plot menunjukkan nilai unik dari variabel "Age", sedangkan sumbu y menunjukkan frekuensi setiap nilai unik.

> Baris "plt.figure(figsize=(20,10))" menetapkan ukuran plot menjadi 20 inci kali 10 inci.

> Garis "plt.xticks(ha='right',rotation='90')" memutar label sumbu x sebesar 90 derajat dan menyejajarkannya ke kanan.

> Baris "sns.barplot(x=pop_values,y=pop_frequencies,color='green')" membuat plot batang dengan "pop_values" pada sumbu x dan "pop_frequencies" pada sumbu y, berwarna hijau.

> Terakhir, kode menggunakan fungsi seaborn "despine" untuk menghapus duri atas dan kanan plot, dan menyetel label judul dan sumbu dengan fungsi "plt.title", "plt.xlabel", dan "plt.ylabel". .

> Secara keseluruhan, kode ini berguna untuk memvisualisasikan distribusi frekuensi variabel kategori seperti "Age" dalam kumpulan data.

```
pop_values, pop_frequencies = np.unique(datas['age'].values.tolist(), return_counts=True)

plt.figure(figsize=(20,10))
plt.xticks(ha="right",rotation="90")
sns.barplot(x=pop_values,y=pop_frequencies,color = 'Lavender')

# Plot formatting
plt.title('Frequency chart of Population - Work Experience Overall distribution',fontsize='14')
sns.despine()
plt.xlabel('Work Experience Overall',fontsize='12')
plt.ylabel('Frequency',fontsize='12')
```

DISTRIBUTION SAMPLING (SALARY)

```
datas = datas[datas["USD Salary"] != 875000.0]
```

```
data_salary = datas['USD Salary']
data_salary.describe()
```

```
count       284.000000
mean     127670.235701
std       59714.099940
min         145.000000
25%       90425.704750
50%      120000.000000
75%      154250.000000
max      590000.000000
Name: USD Salary, dtype: float64
```

```
data_salary.dtype
```

```
dtype('float64')
```

```
data_salary = data_salary.astype(int)
```

```
data_salary.dtype
```

```
dtype('int32')
```

Population Standard deviation

> Kode ini pertama-tama menghitung nilai unik dan frekuensinya untuk variabel "USD Salary" menggunakan fungsi "np.unique" dengan "return_counts=True".
>
> Baris "plt.figure(figsize=(30,20))" menetapkan ukuran plot menjadi 30 inci kali 20 inci.
>
> Garis "plt.xticks(ha='right',rotation='90')" memutar label sumbu x sebesar 90 derajat dan menyejajarkannya ke kanan.
>
> Baris "sns.barplot(x=pop_values,y=pop_frequencies,color='brown')" membuat plot batang dengan "pop_values" pada sumbu x dan "pop_frequencies" pada sumbu y, berwarna cokelat.
>
> Kode kemudian menggunakan fungsi "despine" seaborn untuk menghapus duri atas dan kanan plot, dan menyetel label judul dan sumbu dengan fungsi "plt.title", "plt.xlabel", dan "plt.ylabel".
>
> Terakhir, kode mencetak simpangan baku variabel "USD Salary" dalam kumpulan data "data" menggunakan fungsi "np.std" dan pemformatan string.
>
> Secara keseluruhan, kode ini berguna untuk memvisualisasikan distribusi frekuensi variabel kontinu seperti "USD Salary" dalam kumpulan data dan menghitung standar deviasinya.

```
pop_values, pop_frequencies = np.unique(data_salary, return_counts=True)

plt.figure(figsize=(30,20))
plt.xticks(ha="right",rotation="90")
sns.barplot(x=pop_values,y=pop_frequencies,color = 'brown')

# Plot formatting
plt.title('Frequency chart of Population - Salary distribution',fontsize='14')
sns.despine()
plt.xlabel('Salary',fontsize='12')
plt.ylabel('Frequency',fontsize='12')

print(f'Standard deviation of the population  = {np.std(datas["USD Salary"]):.3f}')
```

3. T-test, ANOVA, Chi-square

```
import numpy as np
import pandas as pd
from scipy.stats import ttest_ind
from scipy.stats import f_oneway
from scipy.stats import chi2_contingency
```

```
#Connect colabs dengan drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# Membaca file csv dan menampilkan dalam bentuk tabel
df = pd.read_excel(r'/content/drive/MyDrive/Kuliah/foundation/Manager Salary Survey 2021-FINAL (2).xlsx')
display(df)
```

	time	age	industry	job_title	need_additional_context	salary	mo
0	2021-04-27 11:02:09.743	25-34	Education (Higher Education)	Research and Instruction Librarian	NaN	55000	
1	2021-04-27 11:02:21.562	25-34	Computing or Tech	Change & Internal Communications Manager	NaN	54600	
2	2021-04-27 11:02:38.125	25-34	Accounting, Banking & Finance	Marketing Specialist	NaN	34000	
3	2021-04-27 11:02:40.643	25-34	Nonprofits	Program Manager	NaN	62000	
4	2021-04-27 11:02:41.793	25-34	Accounting, Banking & Finance	Accounting Manager	NaN	60000	
...	...	...	...	...	...	...	
27869	2023-02-27 06:22:35.240	25-34	Computing or Tech	Sr software engineer	NaN	190000	
27870	2023-03-01 09:43:37.098	45-54	Property or Construction	Property Manager	Senior	55000	
27871	2023-03-06 18:16:18.899	25-34	Behavioral Health	Mental Health Therapist	NaN	52416	
27872	2023-03-09 09:11:48.506	18-24	Computing or Tech	Help Desk Associate Analyst	NaN	36000	
27873	2023-03-12 14:32:04.060	25-34	Recruitment or HR	HR generalist	NaN	60000	

27874 rows × 20 columns

Uji t-test

Hipotesis

- Hipotesis null: Tidak ada perbedaan signifikan antara gaji karyawan pria dan wanita.
- Hipotesis alternatif : Terdapat perbedaan signifikan antara gaji karyawan pria dan wanita

Penjelasan

- Kode di bawah digunakan untuk memisahkan data gaji karyawan laki-laki dan perempuan dari dataset, lalu melakukan uji statistik t-test antara kedua kelompok tersebut.
- data gaji pegawai laki-laki akan disimpan ke dalam variabel male_salary

- data gaji pegawai perempuan akan disimpan ke dalam variabel female_salary
- t_stat adalah nilai statistik uji t, yang menunjukkan seberapa jauh nilai rata-rata kedua kelompok data berbeda dalam satuan standar error.
- p_val (atau nilai p) adalah probabilitas bahwa perbedaan antara rata-rata kedua kelompok data tidak signifikan secara statistik. Semakin kecil nilai p_val, semakin signifikan perbedaan antara rata-rata kedua kelompok data.
- Hasil kode di atas menunjukkan nilai t_stat sebesar 0.8458312715345003 dan nilai p_val sebesar 0.3976579476045651

```
# Memisahkan data gaji karyawan pria dan wanita
male_salary = df[df['gender']=='Man']['salary']
female_salary = df[df['gender']=='Woman']['salary']

# Melakukan uji statistik t-test
t_stat, p_val = ttest_ind(male_salary, female_salary, equal_var=False)

print("t-statistic: ", t_stat)
print("p-value: ", p_val)
```

```
t-statistic:  0.8458312715345003
p-value:  0.3976579476045651
```

Penjelasan

- Kode di bawah digunakan untuk memisahkan data gaji karyawan laki-laki dan perempuan dari dataset, lalu melakukan uji statistik t-test antara kedua kelompok tersebut.
- t_stat adalah nilai statistik uji t, yang menunjukkan seberapa jauh nilai rata-rata kedua kelompok data berbeda dalam satuan standar error.
- p_val (atau nilai p) adalah probabilitas bahwa perbedaan antara rata-rata kedua kelompok data tidak signifikan secara statistik. Semakin kecil nilai p_val, semakin signifikan perbedaan antara rata-rata kedua kelompok data.
- Hasil kode di atas menunjukkan nilai t_stat sebesar 0.8458312715345003 dan nilai p_val sebesar 0.3976579476045651

```
level_of_significance = 0.05 # Tingkat signifikansi

if p_val < level_of_significance:
    print("Hipotesis nol ditolak. Terdapat perbedaan signifikan antara gaji karyawan pria dan wanita.")
else:
    print("Hipotesis nol diterima. Tidak ada perbedaan signifikan antara gaji karyawan pria dan wanita.")
```

```
 Hipotesis nol diterima. Tidak ada perbedaan signifikan antara gaji karyawan pria dan wanita.
```

Anova Test

- Hipotesis null: Tidak ada hubungan antara gender dan kategori gaji pada karyawan dengan pengalaman bekerja.
- Hipotesis alternatif: Terdapat hubungan antara gender dan kategori gaji pada karyawan dengan pengalaman bekerja

Penjelasan Anova Test

- Kode 1 sampai 8 digunakan untuk memisahkan data gaji karyawan berdasarkan pengalaman kerja (work experience) yaitu "1 year or less, 2 - 4 years, 5-7 years, 8 - 10 years, 11 - 20 years, 21 - 30 years, 31 - 40 years, 41 years or more" dan jenis kelamin (gender) laki-laki (man) dan perempuan (woman).
- Dilakukan pengelompokan data gaji berdasarkan kategori work experience dan gender, kemudian disimpan dalam variabel-variabel yang mewakili gaji karyawan pria dan wanita pada setiap kategori work experience. Selanjutnya, variabel-variabel tersebut digunakan sebagai argumen dalam fungsi f_oneway untuk melakukan uji statistik ANOVA.
- f_oneway digunakan untuk melakukan uji statistik ANOVA pada data gaji yang telah dibagi-bagi berdasarkan kategori work experience dan gender.
- Hasil dari kode di atas yaitu F-statistic menghasilkan nilai sebesar 3.485432858154642 dan p-value menghasilkan nilai sebesar 5.149148438764387e-06

```
# Memisahkan data gaji karyawan pria dan wanita untuk setiap tingkat pendidikan tertinggi
#kode1
one_or_less_salary = df[df['work_experience_overall']=="1 year or less"]
male_one_or_less_salary = one_or_less_salary[one_or_less_salary['gender']=='Man']['USD Salary']
female_one_or_less_salary = one_or_less_salary[one_or_less_salary['gender']=='Woman']['USD Salary']

#kode2
two_four_salary = df[df['work_experience_overall']=='2 - 4 years']
male_two_four_salary = two_four_salary[two_four_salary['gender']=='Man']['USD Salary']
```

```
female_two_four_salary = two_four_salary[two_four_salary['gender']=='Woman']['USD Salary']

#kode3
five_to_seven_salary = df[df['work_experience_overall']=='5-7 years']
male_five_to_seven_salary = five_to_seven_salary[five_to_seven_salary['gender']=='Man']['USD Salary']
female_five_to_seven_salary = five_to_seven_salary[five_to_seven_salary ['gender']=='Woman']['USD Salary']

#kode4
eight_to_ten_salary = df[df['work_experience_overall']=='8 - 10 years']
male_eight_to_ten_salary = eight_to_ten_salary[eight_to_ten_salary['gender']=='Man']['USD Salary']
female_eight_to_ten_salary = eight_to_ten_salary[eight_to_ten_salary ['gender']=='Woman']['USD Salary']

#kode5
eleven_to_twenty_salary = df[df['work_experience_overall']=='11 - 20 years']
male_eleven_to_twenty_salary = eleven_to_twenty_salary[eleven_to_twenty_salary['gender']=='Man']['USD Salary']
female_eleven_to_twenty_salary = eleven_to_twenty_salary[eleven_to_twenty_salary ['gender']=='Woman']['USD Salary']

#kode6
twentyone_to_thrity_salary = df[df['work_experience_overall']=='21 - 30 years']
male_twentyone_to_thrity_salary = twentyone_to_thrity_salary[twentyone_to_thrity_salary['gender']=='Man']['USD Salary']
female_twentyone_to_thrity_salary = twentyone_to_thrity_salary[twentyone_to_thrity_salary ['gender']=='Woman']['USD Salary']

#kode7
thrityone_to_fourty_salary = df[df['work_experience_overall']=='31 - 40 years']
male_thrityone_to_fourty_salary = thrityone_to_fourty_salary[thrityone_to_fourty_salary['gender']=='Man']['USD Salary']
female_thrityone_to_fourty_salary = thrityone_to_fourty_salary[thrityone_to_fourty_salary ['gender']=='Woman']['USD Salary']

#kode8
fourtyone_or_more_salary = df[df['work_experience_overall']=='41 years or more']
male_fourtyone_or_more_salary = fourtyone_or_more_salary[fourtyone_or_more_salary['gender']=='Man']['USD Salary']
female_fourtyone_or_more_salary = fourtyone_or_more_salary[fourtyone_or_more_salary ['gender']=='Woman']['USD Salary']

# Melakukan uji statistik ANOVA
f_stat, p_val = f_oneway(male_one_or_less_salary,female_one_or_less_salary,male_two_four_salary,female_two_four_salary,
                         male_five_to_seven_salary,female_five_to_seven_salary, male_eight_to_ten_salary, female_eight_to_ten_salary,
                         male_eleven_to_twenty_salary, female_eleven_to_twenty_salary, male_twentyone_to_thrity_salary, female_twentyone_to_t
                         male_thrityone_to_fourty_salary, female_thrityone_to_fourty_salary, male_fourtyone_or_more_salary,
                         male_fourtyone_or_more_salary)

print("F-statistic: ", f_stat)
print("p-value: ", p_val)
```

```
F-statistic:  3.485432858154642
p-value:  5.149148438764387e-06
```

Penjelasan Anova Test

- level of significance ditetapkan sebagai 0,05, yang berarti jika nilai p yang dihasilkan dari uji statistik ANOVA lebih kecil dari 0,05, hipotesis nol akan ditolak.
- Setelah dibandingkan antara p-value dengan level of significance ternyata hasil p-value lebih kecil dibandingkan dengan level of significance.
- Apabila p-value < level of significance maka dapat disimpulkan bahwa terdapat perbedaan yang signifikan dalam gaji karyawan antara karyawan pria dan wanita pada berbagai tingkat pendidikan tertinggi.
- Item daftar

```
# Define alpha level (significance level)
level_of_significance = 0.05

if p_val < level_of_significance:
    print('Nilai p-value lebih kecil dari level_of_significance. Maka hipotesis nol dapat di tolak.')
    print('Terdapat hubungan antara gender dan kategori gaji pada karyawan dengan pengalaman bekerja.')
else:
    print('Nilai p-value lebih kecil dari level_of_significance. Maka hipotesis nol tidak berhasil di tolak.')
    print('Tidak ada hubungan antara gender dan kategori gaji pada karyawan dengan pengalaman bekerja.')
```

```
Nilai p-value lebih kcil dari level_of_significance. Maka hipotesis nol dapat di tolak.
Terdapat hubungan antara gender dan kategori gaji pada karyawan dengan pengalaman bekerja.
```

Chi-Square test

- Hipotesis nol: Tidak terdapat hubungan antara tingkat pendidikan dan gaji yang diterima.

- Hipotesis Alternatif: Terdapat hubungan antara tingkat pendidikan dan gaji yang diterima.

Penjelasan Chi-Square

- observed adalah variabel yang digunakan untuk membuat tabel kontingensi dari dua variabel, yaitu hights_education_complate dan USD Salary. Variabel pertama (df['hights_education_complate']) dijadikan sebagai baris dan variabel kedua (df['USD Salary']) dijadikan sebagai kolom pada tabel kontingensi.
- chi2 adalah Statistik uji chi-square yang dihasilkan dari fungsi chi2_contingency. Statistik ini akan digunakan untuk menghitung nilai p-value dari uji chi-square.
- p-value adalah nilai hasil uji chi-square yang dihasilkan dari fungsi chi2_contingency. P-value adalah nilai probabilitas bahwa hasil uji chi-square yang diperoleh adalah acak belaka.
- dof (degree of freedom) adalah Derajat kebebasan dari hasil uji chi-square yang dihasilkan dari fungsi chi2_contingency. Derajat kebebasan adalah jumlah variabel dikurangi 1.
- expected adalah Tabel kontingensi yang diharapkan dari kedua variabel yang diuji, jika tidak ada hubungan antara keduanya. Variabel ini dihasilkan dari fungsi chi2_contingency.

```
from scipy.stats import chi2_contingency

# Menyiapkan data untuk uji chi-square
observed = pd.crosstab(df['hights_education_complate'], df['USD Salary'])

# Uji chi-square
chi2, p_value, dof, expected = chi2_contingency(observed)

print("expected: ",expected)
print("dof: ",dof)
print("chi2: ",chi2)
print("p value: ",p_value)

# Cetak hasil uji chi-square
if p_value < 0.05:
    print("Hipotesis H0 ditolak. Terdapat hubungan antara tingkat pendidikan dan gaji yang diterima.")
else:
    print("Hipotesis H0 diterima. Tidak terdapat hubungan antara tingkat pendidikan dan gaji yang diterima.")
```

```
expected:  [[7.28337491 0.48555833 0.48555833 ... 0.48555833 0.48555833 0.48555833]
 [0.3410693  0.02273795 0.02273795 ... 0.02273795 0.02273795 0.02273795]
 [4.78093482 0.31872899 0.31872899 ... 0.31872899 0.31872899 0.31872899]
 [0.7705238  0.05136825 0.05136825 ... 0.05136825 0.05136825 0.05136825]
 [0.71629975 0.04775332 0.04775332 ... 0.04775332 0.04775332 0.04775332]
 [1.10779742 0.07385316 0.07385316 ... 0.07385316 0.07385316 0.07385316]]
dof:  23160
chi2:  27877.397386024673
p value:  1.7277799335975983e-94
Hipotesis H0 ditolak. Terdapat hubungan antara tingkat pendidikan dan gaji yang diterima.
```

4. Linear Regression

Prediksi Salary berdasarkan fitur Age, Work Experience, Gender, dan Education dengan menggunakan Multiple Linear Regression

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from sklearn.feature_selection import RFE
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

```
#Connect colabs dengan drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# Membaca file csv dan menampilkan dalam bentuk tabel
df = pd.read_excel(r'/content/drive/MyDrive/Kuliah/foundation/Manager Salary Survey 2021-FINAL (2).xlsx')
# Software Engineer              286
# Project Manager                230
# Senior Software Engineer       196
# Director                       196
# Program Manager                152
display(df)
```

	time	age	industry	job_title	need_additional_context	salary	monetary_comp
0	2021-04-27 11:02:09.743	25-34	Education (Higher Education)	Research and Instruction Librarian	NaN	55000	
1	2021-04-27 11:02:21.562	25-34	Computing or Tech	Change & Internal Communications Manager	NaN	54600	
2	2021-04-27 11:02:38.125	25-34	Accounting, Banking & Finance	Marketing Specialist	NaN	34000	
3	2021-04-27 11:02:40.643	25-34	Nonprofits	Program Manager	NaN	62000	
4	2021-04-27 11:02:41.793	25-34	Accounting, Banking & Finance	Accounting Manager	NaN	60000	
...	...	...	...	...	...	...	
27869	2023-02-27 06:22:35.240	25-34	Computing or Tech	Sr software engineer	NaN	190000	
27870	2023-03-01 09:43:37.098	45-54	Property or Construction	Property Manager	Senior	55000	
27871	2023-03-06 18:16:18.899	25-34	Behavioral Health	Mental Health Therapist	NaN	52416	
27872	2023-03-09 09:11:48.506	18-24	Computing or Tech	Help Desk Associate Analyst	NaN	36000	
27873	2023-03-12 14:32:04.060	25-34	Recruitment or HR	HR generalist	NaN	60000	

27874 rows × 20 columns

Kode di bawah untuk mendapatkan deskripsi statistik dari kolom 'USD Salary'. Deskripsi statistik yang diperoleh meliputi jumlah data, rata-rata, standar deviasi, nilai minimum, kuartil pertama, median, kuartil ketiga, nilai maksimum dari data.

```
df['USD Salary'].describe()
# df['USD Salary'].max()
```

```
count    2.787400e+04
mean     9.106027e+04
std      6.145100e+05
min      0.000000e+00
25%      5.300000e+04
50%      7.400000e+04
75%      1.050526e+05
max      1.020000e+08
Name: USD Salary, dtype: float64
```

```
# # Disini kita menghilangkan 1 outliner
df = df[df["USD Salary"] != 102000000.0]
df = df[df["USD Salary"] != 2111538.0]
# df = df[df["USD Salary"] != 1250000.0]

# #teradpat 1 outlier
# len(df)
```

```
#menghitung jumlah job title yang unik (distinct)

len(df['job_title'].unique())
```

```
14262
```

- kode di bawah untuk mengubah nilai kategorikal dari kolom job_title menjadi nilai numerik yang sesuai untuk dapat diolah lebih lanjut
- method value_counts(), digunakan untuk menghitung jumlah kemunculan setiap nilai unik pada kolom job_title. Kemudian, nilai-nilai tersebut akan diurutkan berdasarkan jumlah kemunculan dari yang paling sering muncul hingga yang paling jarang.
- selanjutnya membuat dictionary job_dict yang berisi pasangan key-value, dimana key adalah nilai kategorikal dari job_title dan value adalah integer yang merepresentasikan nilai tersebut
- terakhir membuat kolom baru job_title_int yang akan berisi nilai integer yang merepresentasikan nilai kategorikal pada kolom job_title dengan menggunakan method map() untuk memetakan setiap nilai kategorikal pada job_title ke nilai integer yang sesuai pada job_dict.

```
# Create a dictionary mapping each job title to an integer value based on their frequency
job_counts = df['job_title'].value_counts()
job_dict = {job: idx for idx, job in enumerate(job_counts.index)}

# Map the job titles to their corresponding integer values
df['job_title_int'] = df['job_title'].map(job_dict)
```

- Kode di bawah digunakan untuk mengubah nilai-nilai kategori dalam kolom "industry" menjadi bilangan bulat dengan cara membuat kamus (dictionary) yang memetakan masing-masing nilai kategori menjadi bilangan bulat berdasarkan frekuensi kemunculan dari nilai kategori tersebut.
- melakukan penghitungan frekuensi kemunculan dari setiap nilai kategori dalam kolom "industry" dengan menggunakan method value_counts(). Hasil perhitungan akan disimpan dalam variabel job_counts.
- kemudian membuat kamus (job_dict) yang akan memetakan setiap nilai kategori dalam kolom "industry" menjadi bilangan bulat berdasarkan indeksnya pada series job_counts. Indeks pada job_counts menunjukkan urutan frekuensi kemunculan dari nilai kategori yang berbeda dalam kolom "industry".
- selanjutntya, kamus job_dict digunakan untuk mengubah nilai-nilai kategori dalam kolom "industry" menjadi bilangan bulat dan disimpan dalam kolom baru dengan nama "industry_int" menggunakan method map()

```
# Create a dictionary mapping each job title to an integer value based on their frequency
job_counts = df['industry'].value_counts()
job_dict = {job: idx for idx, job in enumerate(job_counts.index)}

# Map the job titles to their corresponding integer values
df['industry_int'] = df['industry'].map(job_dict)
```

```
#menghapus baris atau data pada kolom country_clean yang memiliki nilai NaN atau null
df = df.dropna(subset=["country_clean"])
```

- kode di bawah untuk membuat kamus yang memetakan setiap negara ke nilai integer yang dalam kolom 'country_clean'.
- fungsi 'value_counts()' pada kolom 'country_clean' untuk menghitung frekuensi kemunculan setiap negara dalam kolom tersebut.
- membuat kamus untuk memetakan setiap negara ke nilai integer yang sesuai dengan masing-masing negaranya dalam kolom baru 'country_clean_int' menggunakan fungsi 'map()'.

```
# Create a dictionary mapping each job title to an integer value based on their frequency
job_counts = df['country_clean'].value_counts()
job_dict = {job: idx for idx, job in enumerate(job_counts.index)}

# Map the job titles to their corresponding integer values
df['country_clean_int'] = df['country_clean'].map(job_dict)
```

```
# Menghitung jumlah nilai dalam kolom gender
df['gender'].value_counts()
```

```
Woman    20443
Man       5149
?          983
Name: gender, dtype: int64
```

kode di bawah digunakan untuk melakukan cleaning data. dimana apabila pada kolom gender dan hights_education_complate terdapat baris yang kosong maka akan diisi dengan unknown. serta apabila terdapat baris pada kolom gender yang berisi '?' akan diganti dengan unknown.

```
#mengisi kolom yang NA
df['gender'] = df['gender'].fillna('unknown')
df['gender'] = df['gender'].replace('?', 'unknown')

df['hights_education_complate'] = df['hights_education_complate'].fillna('unknown')
df['gender'].value_counts()
```

```
Woman      20443
Man         5149
unknown     1144
Name: gender, dtype: int64
```

Kode di bawah digunakan untuk melakukan konversi terhadap data dalam kolom age

- baris yang berisi nilai '65 or over' akan dikonversi ke dalam nilai integer 66
- baris yang berisi nilai 'under 18' akan dikonversi kedalam nilai 0

```
# Konversi nilai string ke numerik
def convert_age(age):
    if age == '65 or over':
        return 66
    elif age == 'under 18':
        return 0
    else:
        return int(age.split('-')[0])
df['age_number'] = df['age'].apply(convert_age) # Ambil nilai pertama dari rentang umur
```

kode di bawah digunakan untuk membuat kamus yang merepresentasikan jumlah tahun pengalaman kerja untuk setiap rentang nilai, kamus ini kemudian digunakan untuk mengubah nilai dalam kolom work_experience_overall menjadi numerik dengan menggunakan fungsi apply dan lambda

```
# Buat dictionary yang merepresentasikan jumlah tahun pengalaman kerja untuk setiap rentang nilai
exp_dict = {'1 year or less': 1, '2 - 4 years': 3, '5-7 years': 6, '8 - 10 years': 9,
            '11 - 20 years': 15, '21 - 30 years': 25, '31 - 40 years': 35,
            '41 years or more': 41}

# Ubah nilai work_experience_overall ke dalam nilai numerik
df['work_experience_overall_number'] = df['work_experience_overall'].apply(lambda x: exp_dict[x])
```

kode di bawah digunakan untuk membuat kamus yang merepresentasikan jumlah tahun pengalaman kerja untuk setiap rentang nilai, kamus ini kemudian digunakan untuk mengubah nilai dalam kolom work_experience_field menjadi numerik dengan menggunakan fungsi apply dan lambda

```
# Buat dictionary yang merepresentasikan jumlah tahun pengalaman kerja untuk setiap rentang nilai
exp_dict = {'1 year or less': 1, '2 - 4 years': 3, '5-7 years': 6, '8 - 10 years': 9,
            '11 - 20 years': 15, '21 - 30 years': 25, '31 - 40 years': 35,
            '41 years or more': 41}

# Ubah nilai work_experience_field ke dalam nilai numerik
df['work_experience_field_number'] = df['work_experince_field'].apply(lambda x: exp_dict[x])
```

kode di bawah digunakan untuk membuat kamus yang merepresentasikan jumlah pendidikan tertinggiuntuk setiap rentang nilai, kamus ini kemudian digunakan untuk mengubah nilai dalam kolom hights_education_complate menjadi numerik dengan menggunakan fungsi apply dan lambda

```
# Buat dictionary mapping untuk mengkonversi nilai education
edu_mapping = {"Some college": 0, "High School": 1, "College degree": 2,
               "Master's degree": 3, "Professional degree (MD, JD, etc.)": 4,
               "PhD": 5, "unknown": 6}

# Gunakan method map() untuk mengkonversi nilai education ke integer
df['education_number'] = df['hights_education_complate'].map(edu_mapping).astype(int)
```

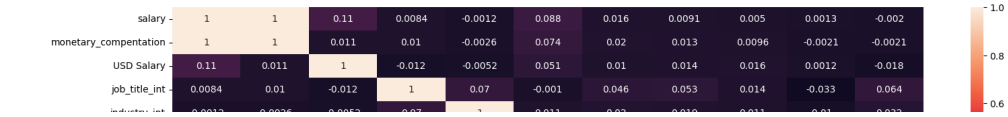
- kode di bawah digunakan untuk menambah baris kosong dalam kolom gender dengan nilai 'unknown', kemudian mereplace nilai'?' menjadi unknown
- membuat kamus utnuk mengkonversi nilai gender ke dalam nilai numerik

```
df['gender'] = df['gender'].fillna('unknown')
df['gender'] = df['gender'].replace('?', 'unknown')

# Buat dictionary mapping untuk mengkonversi nilai gender
gender_mapping = {"unknown": 0, "Man": 1, "Woman": 2}

# Gunakan method map() untuk mengkonversi nilai gender ke integer
df['gender_angka'] = df['gender'].map(gender_mapping).astype(int)
```

Kode di bawah digunakan untuk membuat heatmap korelasi

```
#melihat korelasi data dengan heatmap, dan data yang dapat dilihat korelasoinya hanya data bertipe int
plt.figure(figsize=(18,5))
sns.heatmap(df.corr(), annot=True)
#pada gambar dibawah dapat dilihat korelasi tertinggi yang mempengaruhi salary
```

```
The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will de
<Axes: >
```

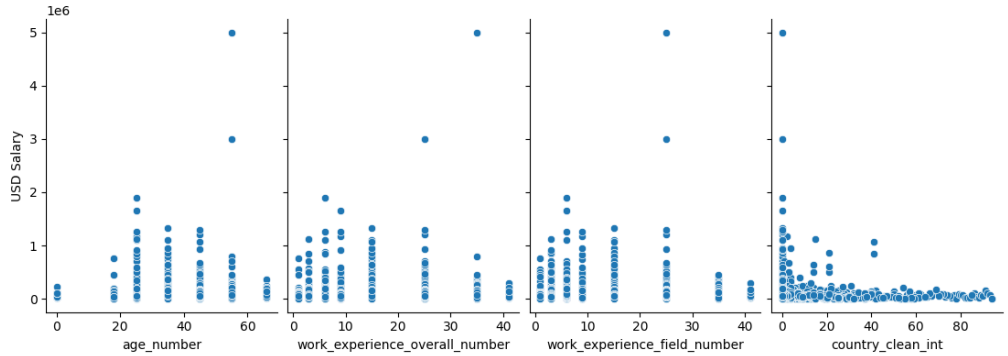



```
# df = df[(df["job_title"]== "Software Engineer")]
```

Kode di bawah digunakan untuk membuat pairplot (scatterplot matrix) yang menampilkan hubungan antara age_number, work_experience_overall_number, work_experience_field_number, country_clean_int dengan variabel target yaitu yaitu "USD Salary"

```
# melihat korelasi future terhadap target apakah linaer atau tidak
# pada hasil korelasi dibawah dengan menggunakan scater plot
sns.pairplot(data=df , x_vars=['age_number','work_experience_overall_number','work_experience_field_number', 'country_clean_int'], y_vars=["U
#sns.show()
```

```
/usr/local/lib/python3.9/dist-packages/seaborn/axisgrid.py:2095: UserWarning: The `size` parameter
  warnings.warn(msg, UserWarning)
<seaborn.axisgrid.PairGrid at 0x7fa21dc30a60>
```




```
#membagi data future dengan data target
X = df.loc[:,['age_number','work_experience_overall_number','work_experience_field_number', 'country_clean_int']]
y = df.loc[:,['USD Salary']]
#X sebagai feature = independent variable = variable bebas
#y sebagai label / Target = dependent variable = variable terikat
print(X)
print(y)
```

```
       age_number  work_experience_overall_number  \
0              25                               6
1              25                               9
2              25                               3
3              25                               9
4              25                               9
...           ...                             ...
27869          25                               6
27870          45                              25
27871          25                               3
27872          18                               6
27873          25                               1

       work_experience_field_number  country_clean_int
0                                 6                  0
1                                 6                  2
2                                 3                  0
3                                 6                  0
4                                 6                  0
...                             ...                ...
27869                             6                  0
27870                            15                  0
27871                             3                  0
27872                             3                  0
27873                             1                  0

[26736 rows x 4 columns]
```

```
            USD Salary
      0       55000.0000
      1       67786.6098
      2       34000.0000
      3       62000.0000
      4       60000.0000
      ...            ...
      27869  190000.0000
      27870   55000.0000
      27871   52416.0000
      27872   36000.0000
      27873   60000.0000

      [26736 rows x 1 columns]
```

```
#standarscaler digunakan agar tidak ada future yang memiliki nilai besar mendominasi mempengaruhi target , maka nilai dari setiap
#future dibuat dalam rentan yang sama
# rumus standar scaler = x - x(mean) / std(x)
#keterangan : x = future
# pada data salary diatas saya menggunakan standarscaller untuk melakukan standarisasi future data salary agar ukuran data future sama rata

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X = scaler.fit_transform(X)
#X_test = scaler.fit_transform(X_test)

print(X)
```

```
     [[-0.77163679 -0.85082371 -0.48816911 -0.17579308]
      [-0.77163679 -0.47425939 -0.48816911  0.28496901]
      [-0.77163679 -1.22738802 -0.91490408 -0.17579308]
      ...
      [-0.77163679 -1.22738802 -0.91490408 -0.17579308]
      [-1.57441942 -0.85082371 -0.91490408 -0.17579308]
      [-0.77163679 -1.4784309  -1.19939407 -0.17579308]]
```

```
# from sklearn.model_selection import train_test_split
# X_train , X_test , y_train , y_test = train_test_split(X, y , test_size=0.1 , random_state=0)
#membagi data train dan data  test dengan menggunakan random test split dari sklearn dengan menggunakan random satate( untuk mengacak data) y

from sklearn.model_selection import train_test_split
X_train , X_test , y_train , y_test = train_test_split(X , y , test_size=0.1 , random_state=42)
print(f'data training feature: {X_train .shape}')
print(f'data testing feature: {X_test.shape}')
print(f'data training target : {y_train.shape}')
print(f'data testing target : {y_test.shape}')
```

```
     data training feature: (24062, 4)
     data testing feature: (2674, 4)
     data training target : (24062, 1)
     data testing target : (2674, 1)
```

```
#membuat model linaer regresion untuk melakukan prediksi data
from sklearn.linear_model import LinearRegression
regresion = LinearRegression()
regresion = regresion.fit(X_train , y_train)
```

```
# pada model regresi linear menggunakan multiple linear regresion

# rumus multiple linear regresion
# y = a +m1*x1 + m2*x2
#keterangan
# a = intercpt / kosntanta (yang mempengaruhi kemiringan garis) (nilai dari Y apabila x benilai =0)
# y = defendent var (target)
# m1 , m2 , .. mn = coefisien regresi (pengaruh positif ataupun negatif)
# jika var coef bernilai 0 maka y = b

print(regresion.coef_)
print(regresion.intercept_)
```

```
     [[  278.21275486 -2067.05571088 16248.56293231 -3181.7433842 ]]
     [88357.5773142]
```

Fungsi cross_val_score akan mengembalikan nilai R-squared pada setiap fold yang kemudian dapat digunakan untuk mengevaluasi kinerja model Linear Regression

```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
cross_val_score(lr,X,y,cv=4)
```

```
array([0.01550086, 0.05495555, 0.04518831, 0.03584001])
```

melakukan evaluasi model regresi linear pada data uji

```
from sklearn.metrics import mean_absolute_error , r2_score, mean_squared_error
y_pred = regresion.predict(X_test)

mse_rrg = mean_squared_error(y_test , y_pred)

print(f'Nilai r2 error = {r2_score(y_test,y_pred)}')
print(f'Nilai Mean Square Error {mse_rrg}')
print(f'Nilai regresion score = {regresion.score(X_test, y_test)}')
```

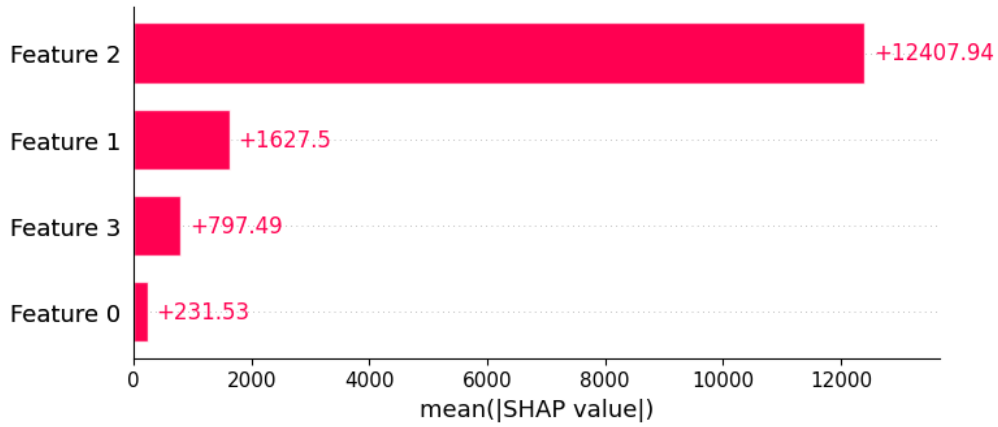
```
Nilai r2 error = 0.06139341213857874
Nilai Mean Square Error 3099457374.8196025
Nilai regresion score = 0.06139341213857874
```

SHAP (SHapley Additive exPlanations) adalah sebuah library pada Python untuk melakukan interpretasi model Machine Learning. Library ini memungkinkan pengguna untuk melihat kontribusi tiap fitur dalam model terhadap prediksi atau output yang dihasilkan oleh model.

```
%pip install shap==0.23.0
%pip install -I shap
```

```
#melihat variable yang memiliki pengaruh penting terhadap target dengan menggunakan shap value
import shap
explainer = shap.Explainer(regresion.predict , X_test)
shap_value = explainer(X_test)
```

```
#melihat pengaruh setiap future terhadap target dengan menggunakan shap value
shap_value
shap.plots.bar(shap_value)
```




```
print(f'Nilai MSE Linear Regresion = {mse_rrg}')
```

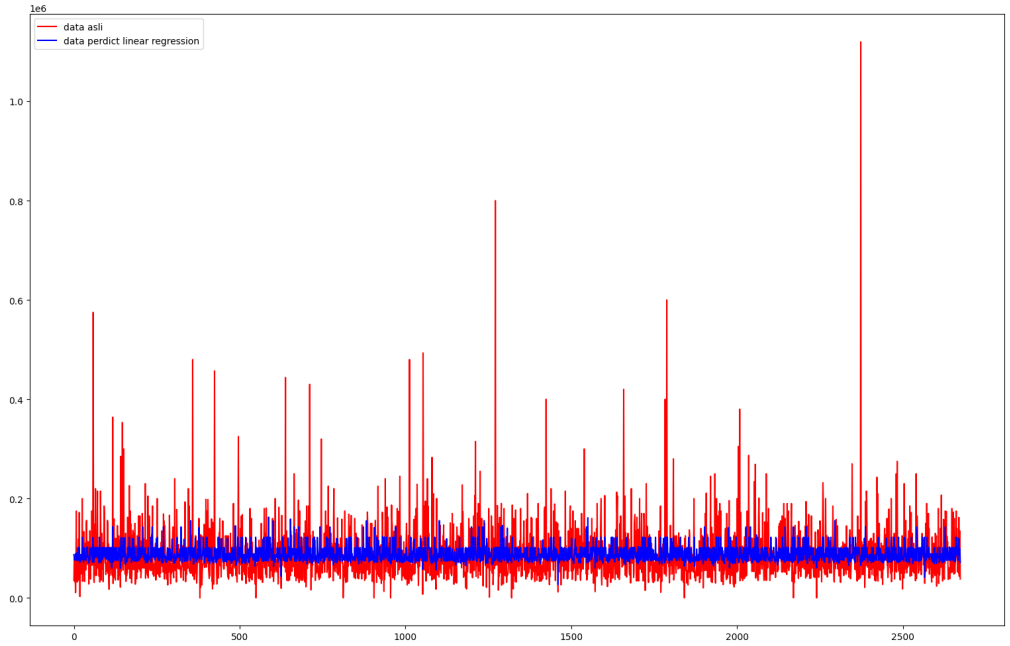
```
Nilai MSE Linear Regresion = 3099457374.8196025
```

Memvisualisasikan hasil prediksi model regresi linier terhadap data asli.

```
y_dtr = pd.Series(np.ravel(y_pred))
plt.figure(figsize=(19,12))
x_data= np.arange(len(X_test))
y_data= y_test.values
plt.plot(x_data , y_data , c='r' ,label='data asli')
plt.plot(x_data , y_dtr , c='b' ,label='data perdict linear regression')
```

```
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7fa16a4af2b0>
```



✓ 0 d selesai pada 20.46