Problem Statement

```
# Scaler is an online tech-versity offering intensive computer science & Data Science courses
# through live classes delivered by tech leaders and subject matter experts. The meticulously
# structured program enhances the skills of software professionals by offering a modern curriculum
# with exposure to the latest technologies. It is a product by InterviewBit.

# You are working as a data scientist with the analytics vertical of Scaler, focused on profiling
# the best companies and job positions to work for from the Scaler database. You are provided with
# the information for a segment of learners and tasked to cluster them on the basis of their job profile,
# company, and other features. Ideally, these clusters should have similar characteristics.
```

Importing libraries / Read data

```
import pandas as pd
import numpy as np
from matplotlib import rcParams
rcParams['figure.figsize'] = 10, 10
import seaborn as sns
import matplotlib.pyplot as plt
```

```
import warnings
warnings.filterwarnings("ignore")
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files scaler_clustering.csv

- **scaler_clustering.csv**(text/csv) - 24735965 bytes, last modified: 9/22/2023 - 100% done

```
Saving scaler_clustering.csv to scaler_clustering (2).csv
```

```
import pandas as pd
import io

df = pd.read_csv(io.BytesIO(uploaded['scaler_clustering (2).csv']))
print(df)
```

```
        Unnamed: 0              company_hash  \
0                0             atrgxnnt xzaxv
1                1  qtrxvzwt xzegwgbb rxbxnta
2                2               ojzwnvwnxw vx
3                3                   ngpgutaxv
4                4                  qxen sqghu
...            ...                        ...
205838      206918                   vuurt xzw
205839      206919                   husqvawgb
205840      206920                    vwwgrxnt
205841      206921             zgn vuurxwvmrt
205842      206922              bgqsvz onvzrtj

                                               email_hash  orgyear      ctc  \
0       6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...   2016.0  1100000
1       b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...   2018.0   449999
2       4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...   2015.0  2000000
3       effdede7a2e7c2af664c8a31d9346385016128d66bbc58...   2017.0   700000
4       6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...   2017.0  1400000
...                                                   ...      ...      ...
205838  70027b728c8ee901fe979533ed94ffda97be08fc23f33b...   2008.0   220000
205839  7f7292ffad724ebbe9ca860f515245368d714c84705b42...   2017.0   500000
205840  cb25cc7304e9a24facda7f5567c7922ffc48e3d5d6018c...   2021.0   700000
205841  fb46a1a2752f5f652ce634f6178d0578ef6995ee59f6c8...   2019.0  5100000
205842  0bcfc1d05f2e8dc4147743a1313aa70a119b41b30d4a1f...   2014.0  1240000

              job_position  ctc_updated_year
0                    Other            2020.0
1       FullStack Engineer            2019.0
2         Backend Engineer            2020.0
3         Backend Engineer            2019.0
4       FullStack Engineer            2019.0
...                    ...               ...
205838                 NaN            2019.0
205839                 NaN            2020.0
205840                 NaN            2021.0
205841                 NaN            2019.0
205842                 NaN            2016.0

[205843 rows x 7 columns]
```

```
pd.set_option('display.max_rows', None)
```

Shape of the data

```
df.shape
```

```
(205843, 7)
```

Number and data types of variables

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205843 entries, 0 to 205842
Data columns (total 7 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   Unnamed: 0        205843 non-null  int64
 1   company_hash      205799 non-null  object
 2   email_hash        205843 non-null  object
 3   orgyear           205757 non-null  float64
 4   ctc               205843 non-null  int64
 5   job_position      153281 non-null  object
 6   ctc_updated_year  205843 non-null  float64
dtypes: float64(2), int64(2), object(3)
memory usage: 11.0+ MB
```

```
df.head()
```

	Unnamed: 0	company_hash	email_hash	orgyear	ctc	job_position	ctc_updated_year
0	0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	1100000	Other	2020.0
1	1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018.0	449999	FullStack Engineer	2019.0
2	2	ojzwnvwnxw vx	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2015.0	2000000	Backend Engineer	2020.0
3	3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	2017.0	700000	Backend Engineer	2019.0
4	4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	2017.0	1400000	FullStack Engineer	2019.0

```
"""remove unnecessary columns"""
df = df.drop(['Unnamed: 0'], axis=1)
```

```
df1 = df.copy()
```

```
df1.shape
```

```
(205843, 6)
```

Exploratory Data Analysis - Visual and Non-visual

Five point summary (Statistical summary) - Numeric variables

```
#Categorical variables and numerical variables
numeric_df1 = df1.select_dtypes(include=[np.number])
categorical_df1 = df1.select_dtypes(exclude=[np.number])
```

```
numeric_df1.describe()
```

	orgyear	ctc	ctc_updated_year
count	205757.000000	2.058430e+05	205843.000000
mean	2014.882750	2.271685e+06	2019.628231
std	63.571115	1.180091e+07	1.325104
min	0.000000	2.000000e+00	2015.000000
25%	2013.000000	5.300000e+05	2019.000000

Five point summary (Statistical summary) - Categorical variables

```
categorical_df1.describe()
```

	company_hash	email_hash	job_position
count	205799	205843	153281
unique	37299	153443	1017
top	nvnv wgzohrnvzwj otqcxwto	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	Backend Engineer
freq	8337	10	43554

Non-Graphical Analysis: Value counts and unique attributes

```
categorical_df1.columns
```

```
Index(['company_hash', 'email_hash', 'job_position'], dtype='object')
```

```
# print("company_hash:", categorical_df1['company_hash'].unique().tolist())
# print("email_hash:", categorical_df1['email_hash'].unique().tolist())
# print("job_position:", categorical_df1['job_position'].unique().tolist())
```

```
for column in categorical_df1[['company_hash', 'email_hash', 'job_position']]:
    print(column.upper(),': ',categorical_df1[column].nunique())
```

```
COMPANY_HASH :  37299
EMAIL_HASH :  153443
JOB_POSITION :  1017
```

Visual Analysis - Univariate analysis

For continuous variable(s): Distplot, countplot, histogram for univariate analysis

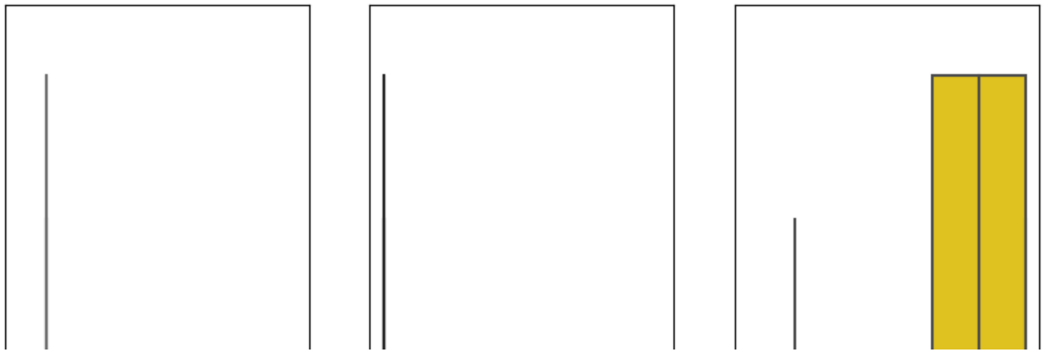
```
numeric_df1.columns
```

```
Index(['orgyear', 'ctc', 'ctc_updated_year'], dtype='object')
```

```
fig, axs = plt.subplots(1, 3, figsize=(10, 7))

sns.boxplot(data=numeric_df1, x="orgyear", color="skyblue", ax=axs[0])
sns.boxplot(data=numeric_df1, x="ctc", color="olive", ax=axs[1])
sns.boxplot(data=numeric_df1, x="ctc_updated_year", color="gold", ax=axs[2])
```

```
<Axes: xlabel='ctc_updated_year'>
```

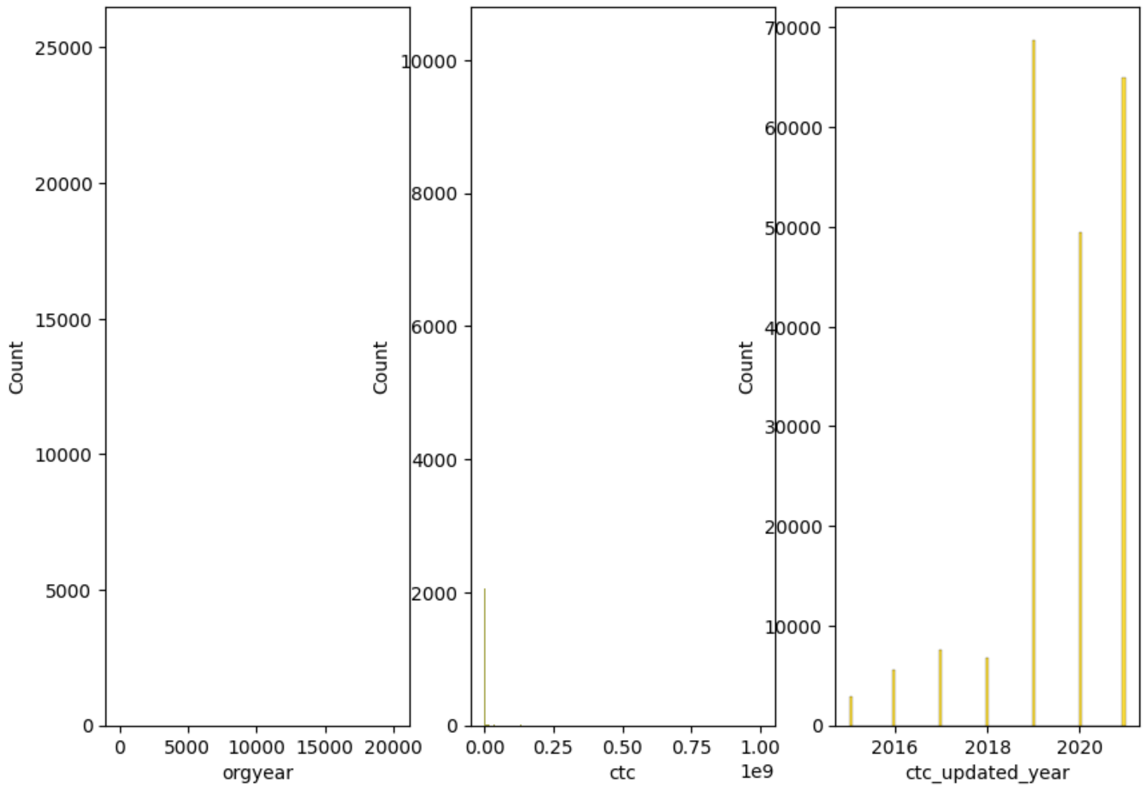


Histogram with distribution curve

```
fig, axs = plt.subplots(1, 3, figsize=(10, 7))

sns.histplot(data=numeric_df1, x="orgyear", color="skyblue", ax=axs[0])
sns.histplot(data=numeric_df1, x="ctc", color="olive", ax=axs[1])
sns.histplot(data=numeric_df1, x="ctc_updated_year", color="gold", ax=axs[2])
```

```
<Axes: xlabel='ctc_updated_year', ylabel='Count'>
```

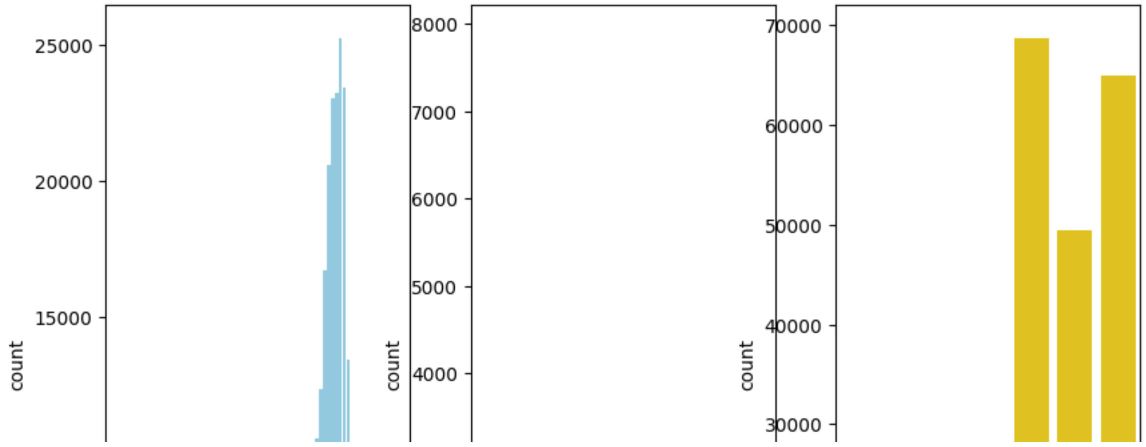


Count plot

```
fig, axs = plt.subplots(1, 3, figsize=(10, 7))

sns.countplot(data=numeric_df1, x="orgyear", color="skyblue", ax=axs[0])
sns.countplot(data=numeric_df1, x="ctc", color="olive", ax=axs[1])
sns.countplot(data=numeric_df1, x="ctc_updated_year", color="gold", ax=axs[2])
```

```
<Axes: xlabel='ctc_updated_year', ylabel='count'>
```



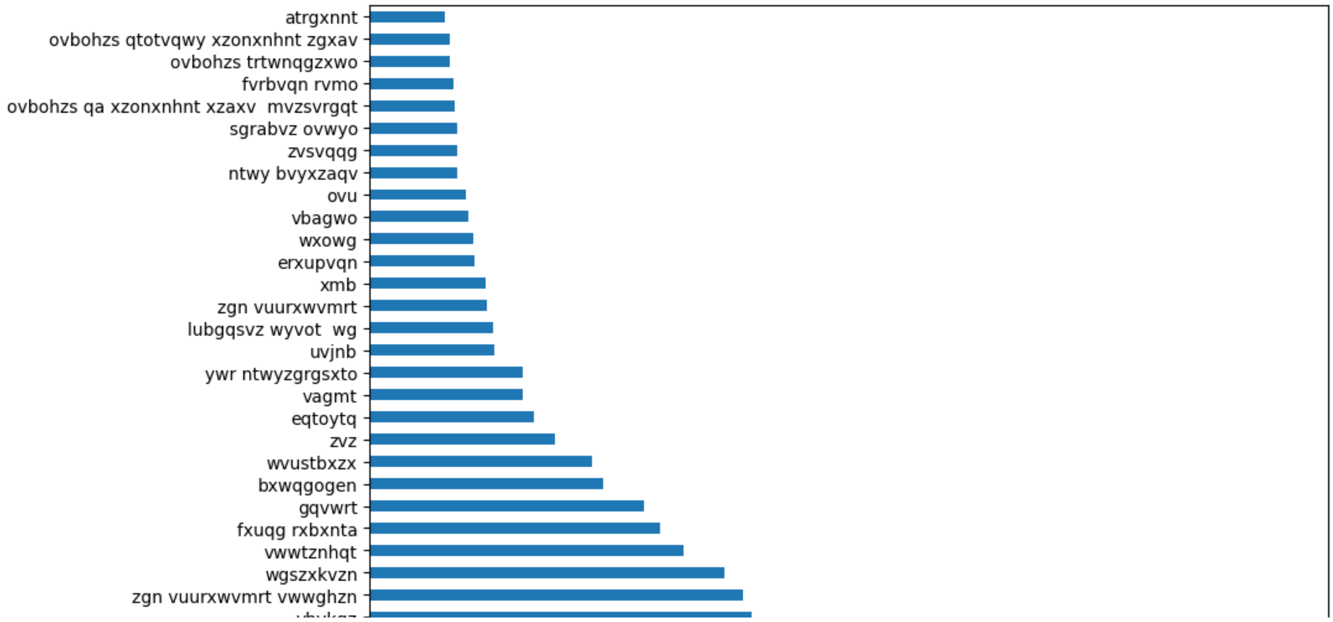
For categorical variable(s): barplot

```
# categorical variales
categorical_df1.columns
```

```
Index(['company_hash', 'email_hash', 'job_position'], dtype='object')
```

```
plt.figure(figsize=(10,7))
df1["company_hash"].value_counts(normalize= True)[:30].sort_values(ascending=False).plot(kind='barh')
```

```
<Axes: >
```




```
plt.figure(figsize=(10,7))
df1["email_hash"].value_counts(normalize= True)[:30].sort_values(ascending=False).plot(kind='barh')
```

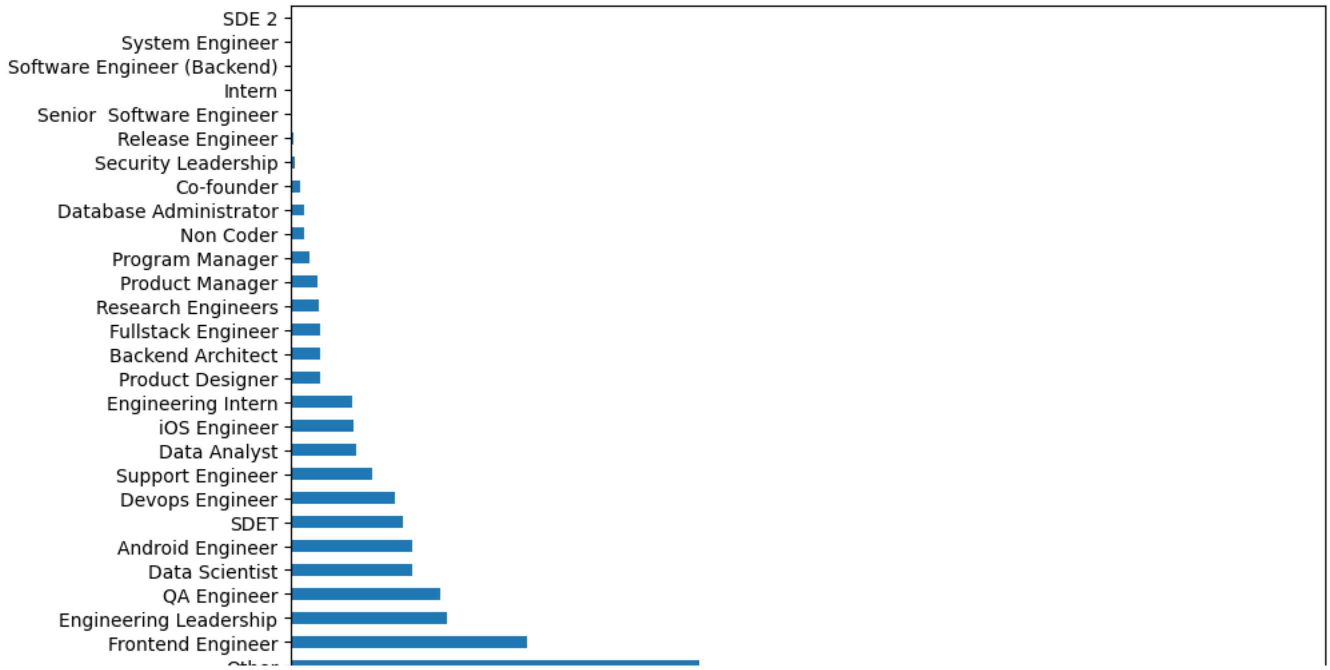
```
<Axes: >
```

```
df1["job_position"].nunique()
```

```
1017
```

```
plt.figure(figsize=(10,7))
df1["job_position"].value_counts(normalize= True)[:30].sort_values(ascending=False).plot(kind='barh')
```

```
<Axes: >
```



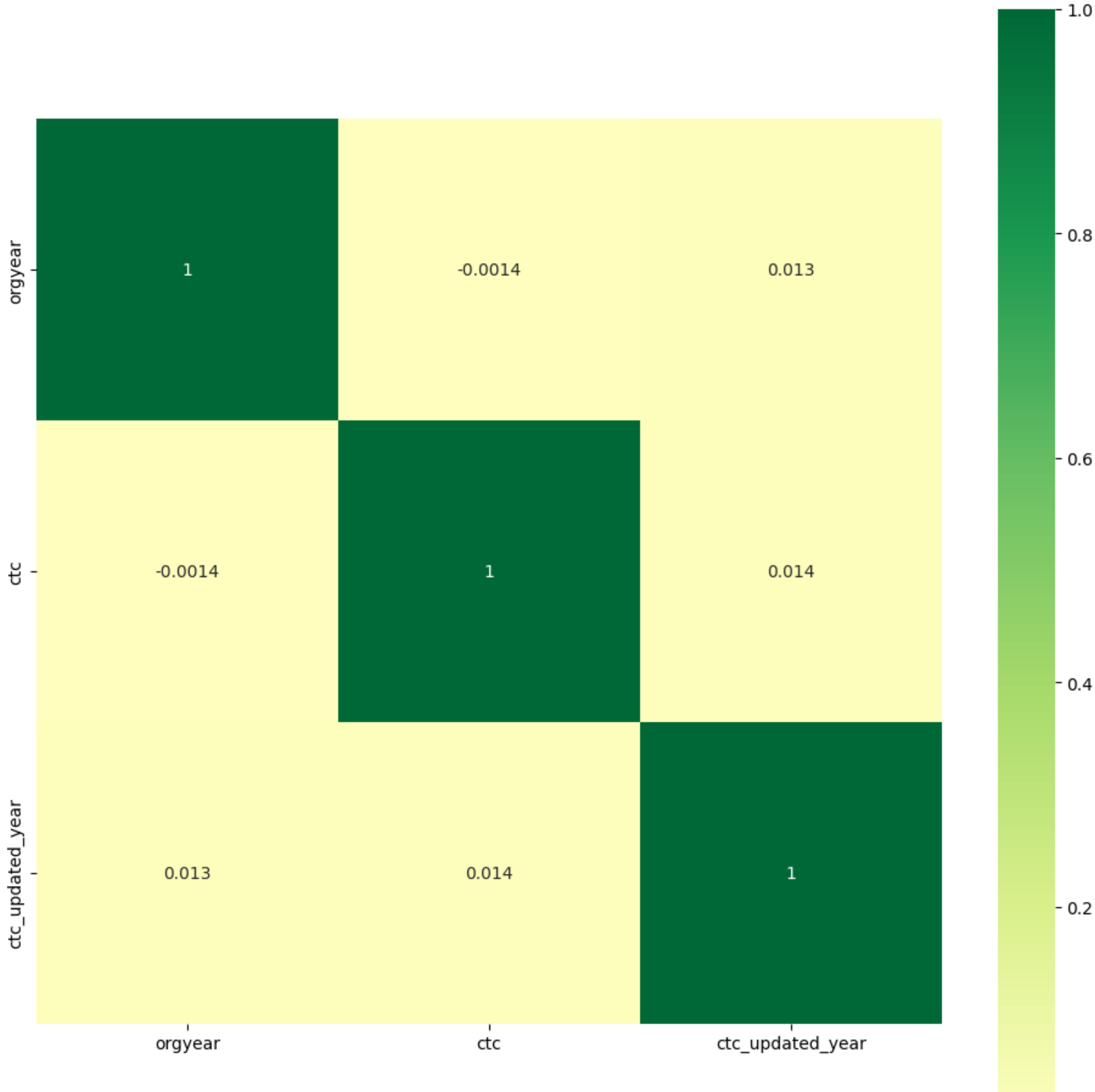

```
sns.histplot(df1["ctc_updated_year"], bins = 5)
```

```
<Axes: xlabel='ctc_updated_year', ylabel='Count'>
```

For correlation: Heatmaps, Pairplots

```
plt.figure(figsize=(12,12))
corr=numeric_df1.corr()
sns.heatmap(corr,square=True,center=0,annot=True,cmap='RdYlGn')
```

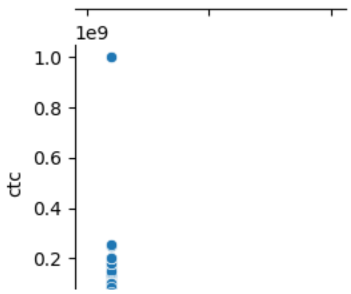
```
<Axes: >
```



Pairplot

```
sns.pairplot(numeric_df1,corner=True)
```

```
<seaborn.axisgrid.PairGrid at 0x7ed4218c33a0>
```




```
df1.columns
```

```
Index(['company_hash', 'email_hash', 'orgyear', 'ctc', 'job_position',
       'ctc_updated_year'],
      dtype='object')
```

Data Preprocessing

```
def Orgyear_fixing(x):
    if x["orgyear"]<=10:
        k = x["ctc_updated_year"]-x["orgyear"]
        return k
    if x["orgyear"] > 2021:
        return 2021
    if x["orgyear"]>=200 and x["orgyear"]<=202:
        return x["orgyear"]*10
    if x["orgyear"]==206.0:
        return 2006
    if x["orgyear"]==209.0:
        return 2009
    if x["orgyear"]==208.0:
        return 2008
    if x["orgyear"] == 91.0:
        return 1991
    if x["orgyear"] == 83.0:
        return 1983
    if x["orgyear"] == 38.0:
        return 2021
    if x["orgyear"] == 1900.0:
        return x["ctc_updated_year"]
    else:
        return x["orgyear"]
```

```
df1["orgyear"] = df1.apply(Orgyear_fixing , axis = 1)
```

```
df1["orgyear"].shape
```

```
(205843,)
```

```
## Checking for missing values
```

```
df1.isnull().sum()
```

```
company_hash          44
email_hash             0
orgyear               86
ctc                    0
job_position       52562
ctc_updated_year       0
dtype: int64
```

```
df1 = df1.dropna(inplace=False)
```

```
df1.isnull().sum()
```

```
company_hash        0
email_hash          0
orgyear             0
ctc                 0
job_position        0
ctc_updated_year    0
dtype: int64
```

```
## remove duplicate rows
```

```
df1.duplicated().sum()
```

```
18
```

```
df1 = df1.drop_duplicates(keep="first", inplace=False)
```

```
df1.duplicated().sum()
```

```
0
```

```
# removing outliers
```

```
df1 = df1[(df1["ctc"]>df1["ctc"].quantile(0.025))&(df1["ctc"]<df1["ctc"].quantile(0.975))]
```

```
df1["job_position"]=df1["job_position"].str.lower().str.replace(r"([^A-Za-z\s])|(\si+$)|(senior\s+)|(development\s+)|(associate\s+)|(memb
```

```
jobs=dict(df1["job_position"].value_counts()[:36]); jobs
```

```
{'backend engineer': 41140,
 'fullstack engineer': 24444,
 'other': 16758,
 'frontend engineer': 9993,
 'qa engineer': 6375,
 'engineering leadership': 6137,
 'android engineer': 5114,
 'data scientist': 5111,
 'sdet': 4881,
 'devops engineer': 4466,
 'support engineer': 3431,
 'data analyst': 2733,
 'ios engineer': 2612,
 'engineering intern': 2570,
 'product designer': 1286,
 'backend architect': 1156,
 'research enginee': 1120,
 'product manager': 1047,
 'program manager': 767,
 'non coder': 573,
 'database administrator': 505,
 'cofounder': 324,
 'software engineer': 324,
 'security leadership': 124,
 'release engineer': 111,
 'sde': 93,
 'system engineer': 73,
 'engineer': 53,
 'consultant': 51,
 'intern': 42,
 'technical staff': 32,
 'software developer': 30,
 'student': 23,
 'data engineer': 20,
 'research engineer': 20,
 'project engineer': 19}
```

```
df1["job_position"] = df1["job_position"].apply(lambda x: "other" if x not in jobs else x)
```

```
df1["job_position"].value_counts()
```

```
backend engineer          41140
fullstack engineer        24444
other                     17788
frontend engineer          9993
qa engineer                6375
engineering leadership     6137
android engineer           5114
data scientist             5111
sdet                       4881
```

```
devops engineer           4466
support engineer          3431
data analyst              2733
ios engineer              2612
engineering intern        2570
product designer          1286
backend architect         1156
research enginee          1120
product manager           1047
program manager            767
non coder                  573
database administrator     505
software engineer          324
cofounder                  324
security leadership        124
release engineer           111
sde                         93
system engineer             73
engineer                    53
consultant                  51
intern                      42
technical staff             32
software developer          30
student                     23
data engineer               20
research engineer           20
project engineer            19
Name: job_position, dtype: int64
```

Check for missing values and Prepare data for KNN Imputation

[ ] ↳ *10 cells hidden*

Feature Engineering

```
df2 = df1.copy()
```

```
#Total Years of experience
df2["TYOE"] = 2023 - df2["orgyear"]
# df3["Exp After ctc update"] = 2023 - df3["ctc_updated_year"]
```

Manual Clustering

```
#Getting the 5 point summary of CTC (mean, median, max, min, count etc) on the basis of Company, Job Position, Years of Experience
```

Analysis on Company , Job Position , TYOE

```
df2.groupby(["company_hash"]).aggregate({"ctc":["mean","median","max","min","count"]}).sort_values(by=("ctc","
```

	ctc				
	mean	median	max	min	count
company_hash					
nvnv wgzohrnvzwj otqcxwto	5.920856e+05	450000.0	5000000	103000	5162
xzegojo	6.305944e+05	500000.0	5000000	105000	3347
vbvkgz	2.128751e+06	2000000.0	5400000	105000	2387
wgszxkvzn	7.380802e+05	600000.0	5400000	102000	2028
gqvwrt	1.550486e+06	1300000.0	5400000	110000	1902
zgn vuurxwvmrt vwwghzn	8.591699e+05	600000.0	5300000	105000	1849
vwwtznhqt	7.846996e+05	620000.0	5380000	114000	1799
fxuqg rxbxnta	6.804468e+05	540000.0	5220000	101000	1663
bxwqgogen	2.520601e+06	2500000.0	5405000	105000	1520
wvustbxzx	7.525612e+05	620000.0	5000000	125000	1315

```
df2.groupby(["job_position"]).aggregate({"ctc":["mean","median","max","min","count"]}).sort_values(by=("ctc","
```

	ctc				
	mean	median	max	min	count
job_position					
backend engineer	1.440039e+06	1200000.0	5450000	101000	41140
fullstack engineer	1.204076e+06	960999.5	5405000	101000	24444
other	9.521868e+05	669500.0	5400000	100800	17788
frontend engineer	1.092837e+06	900000.0	5400000	101000	9993
qa engineer	9.483041e+05	700000.0	5480000	105000	6375
engineering leadership	2.444910e+06	2500000.0	5450000	101000	6137
android engineer	1.128393e+06	900000.0	5200000	102000	5114
data scientist	1.403800e+06	1200000.0	5400000	102000	5111
sdet	1.075060e+06	780000.0	5300000	105000	4881
devops engineer	1.241747e+06	1000000.0	5400000	101000	4466

```
df2.groupby(["TYOE"]).aggregate({"ctc":["mean","median","max","min","count"]}).sort_values(by=("ctc","count"),
```

	ctc				
	mean	median	max	min	count
TYOE					
5.0	1.012581e+06	750000.0	5425000	102000	18139
6.0	1.054489e+06	800000.0	5450000	101000	16973
7.0	1.131329e+06	880000.0	5480000	101000	16901
8.0	1.199006e+06	950000.0	5480000	101000	15404
4.0	9.196483e+05	700000.0	5450000	101000	13411
9.0	1.294848e+06	1000000.0	5420000	104000	12676
10.0	1.427547e+06	1200000.0	5400000	102000	9403
11.0	1.551504e+06	1350000.0	5405000	101000	8065
3.0	9.030894e+05	680000.0	5200000	110000	6634
12.0	1.665518e+06	1500000.0	5400000	101000	6049

```
def q50(x):
    return x.quantile(0.50)
def q75(x):
    return x.quantile(0.75)
```

```
# creating feature/column: designation
```

```
df2_designation = df2.groupby(["job_position","TYOE"]).aggregate({"ctc":[q50,q75]})
```

```
df2_designation.shape
```

```
(820, 2)
```

```
df2_designation.head()
```

		ctc	
		q50	q75
job_position	TYOE		
android engineer	**2.0**	840000.0	1200000.0
	3.0	720000.0	1250000.0
	4.0	800000.0	1100000.0
	5.0	700000.0	1100000.0
	6.0	700000.0	1100000.0

```
dict_designation = {}
for i in range(len(df2_designation.index)):
    dict_designation[df2_designation.index[i]]=df2_designation.values[i]
```

```
df2["Designation"]=df2[["job_position","TYOE","ctc"]].apply(lambda x: 3 if x["ctc"]<dict_designation[(x["job_position"],x["TYOE"])][0] el
```

```
df2.shape
```

```
(144588, 8)
```

```
## creating feature/column: Class
```

```
df2_class = df2.groupby(["company_hash","job_position"]).aggregate({"ctc":[q50,q75]})
```

```
dict_class = {}
for i in range(len(df2_class.index)):
    dict_class[df2_class.index[i]]=df2_class.values[i]
```

```
df2["Class"]=df2[["company_hash","job_position","ctc"]].apply(lambda x: 3 if x["ctc"]<dict_class[(x["company_hash"],x["job_position"])][0
```

```
## creating feature/column: Tier
```

```
ctc_q70 = df2["ctc"].quantile(0.7); ctc_q70
```

```
1500000.0
```

```
ctc_q90 = df2["ctc"].quantile(0.90); ctc_q90
```

```
2650000.0
```

```
df2["Tier"] = df2["ctc"].apply(lambda x: 3 if x<ctc_q70 else(1 if x>ctc_q90 else 2))
```

```
df2["Tier"].value_counts(normalize=True)
```

```
3    0.686316
2    0.214333
1    0.099351
Name: Tier, dtype: float64
```

```
## Top 10 employees (earning more than most of the employees in the company) - Tier 1
```

```
df2[(df2["Tier"]==1)&(df2["Class"]==1)][["email_hash","ctc","Designation","Class","Tier"]].sort_values(by="ctc
```

	email_hash	ctc	Designation	Class	Tier
190719	2f7b5dac85824affd76f79c9d6b0e935daa247f014f4a3...	5450000	1	1	1
182246	2f7b5dac85824affd76f79c9d6b0e935daa247f014f4a3...	5450000	1	1	1
123124	a874bfe165badc8de35ae909725301c5a2cff56fc7c089...	5405000	1	1	1
124031	767fc488187e501dd3325322d74a4c22713aea3833a6d6...	5400000	1	1	1
116912	afbb91f438923174572e560fc394097f66a0a8f56cb8d5...	5400000	1	1	1
13263	4d0e1146c22180610ce3d72172d4fbfc27c7b0e4b94edf...	5400000	1	1	1
130159	a8686ee950d077e3169c52eb677d338256e0992e9fac17...	5400000	1	1	1
126718	79b5707c9a29e28960d4983816318229132edd03d174e5...	5400000	1	1	1
75692	73a5b08f412c85d7710295a9eee7c6e32d2825a2e20c95...	5400000	1	1	1
186267	499a7ff9c81954d615bbb454d31c78c7ffccdb6506ea5c...	5400000	1	1	1

```
## Bottom 10 employees (earning less than most of the employees in the company)- Tier 3
```

```
df2[df2["Tier"]==3][["email_hash","ctc","Designation","Class","Tier"]].sort_values(by="ctc", ascending=True)[:10]
```

	email_hash	ctc	Designation	Class	Tier
155249	dfea01f2c9b0030633005c0d95bc2f93911cd88f98142d...	100800	3	3	3
134906	790aefabf34038e871a0a36337fa9c3bb41545a998532f...	101000	3	3	3
195638	942965a32ef51e2d3fa5fd198fd86ec4dad6910afb084b...	101000	3	3	3
66100	f9f15bc2eb6f1f1e5cb7668684f0862aae2a6fe1bdb5b6...	101000	3	3	3
111627	3de01500c2ef7dd9264fc870f3c509290cf7c8073be5c5...	101000	3	3	3
123123	09bf0444be133ac6d10d2c3b2ab1859ce4663951bcb324...	101000	3	3	3

```
## Top 10 companies (based on their CTC)
```

```
df2[df2["Tier"]==1].groupby(["company_hash"]).aggregate({"ctc":"median"}).sort_values(by="ctc", ascending=Fals
```

company_hash	ctc
erhd vhng	5480000.0
xzexzxnj rv	5480000.0
gw2	5450000.0
uqtwxej	5450000.0
xatvrg	5440000.0
lhznqvd ogrhnxgzo ucn rna	5425000.0
vao ogrhnxgzo uqxcvnt rxbxnta	5420000.0
zvnxgzvr wgbbtqwxvr mvzp lvbvxwv rna	5400000.0
zhbmtqk	5400000.0
eggabgzp	5400000.0

```
## Top 2 positions in every company (based on their CTC)
```

```
df2.drop(columns="email_hash").groupby(["company_hash","job_position"]).aggregate({"ctc":"median"}).sort_value
```

company_hash	job_position	ctc
erhd vhng	product designer	5480000.0
xzexzxnj rv	qa engineer	5480000.0
gw2	cofounder	5450000.0
uqtwxej	program manager	5450000.0
xatvrg	engineering leadership	5440000.0
lhznqvd ogrhnxgzo ucn rna	backend engineer	5425000.0
vao ogrhnxgzo uqxcvnt rxbxnta	qa engineer	5420000.0
owxtzvunxw ojontbo	backend architect	5400000.0
zhbmtqk	product manager	5400000.0
ofgg	ios engineer	5400000.0

```
df2.groupby(["job_position"]).aggregate({"ctc":"median"}).sort_values(by="ctc",ascending=False)
```

job_position	ctc
backend architect	2600000.0
program manager	2500000.0
engineering leadership	2500000.0
sde	1900000.0
research engineer	1800000.0
technical staff	1800000.0
product manager	1650000.0
software engineer	1300000.0
data scientist	1200000.0
consultant	1200000.0
cofounder	1200000.0
research enginee	1200000.0
backend engineer	1200000.0
software developer	1100000.0
engineer	1100000.0
security leadership	1080000.0
data engineer	1065000.0
product designer	1050000.0
devops engineer	1000000.0
fullstack engineer	960999.5
ios engineer	917499.5
release engineer	900000.0
android engineer	900000.0
frontend engineer	900000.0
intern	825000.0
engineering intern	800000.0
sdet	780000.0
qa engineer	700000.0

```
tf=df2.groupby(["company_hash","job_position"]).aggregate({"ctc":"median"}).sort_values(by=["company_hash","ctc"], ascending=[True,False]
tf[:20]
```

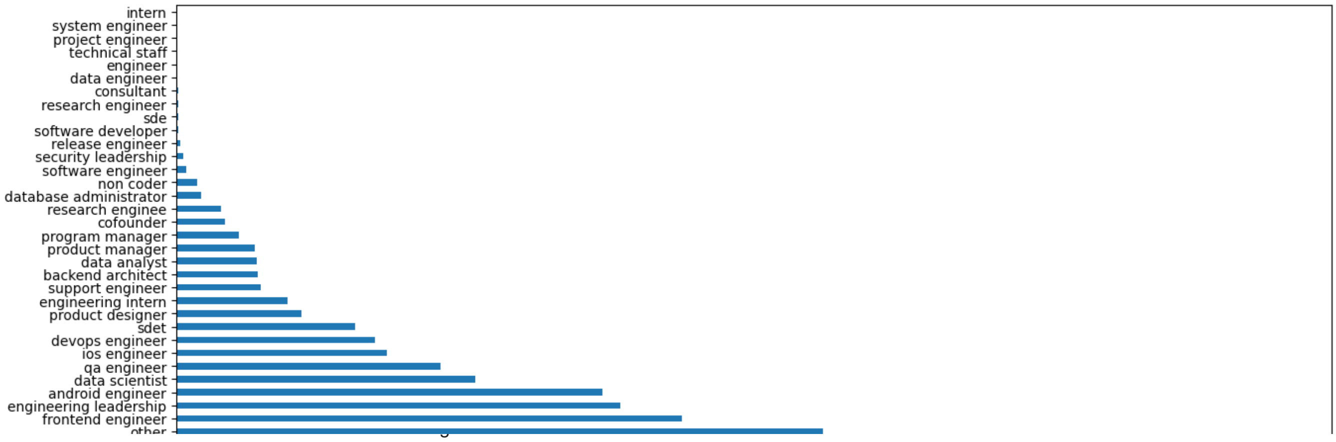
	company_hash	job_position	ctc
0	0000	other	300000.0
1	01 ojztqsj	frontend engineer	830000.0
2	01 ojztqsj	android engineer	270000.0
4	1	other	250000.0

```
tdf=tf.groupby(["company_hash"]).aggregate({"job_position":lambda x: list(x), "ctc":lambda x: list(x)}).reset_index()
```

```
tdf["top_first"]=tdf.apply(lambda x: x["job_position"][0], axis=1)
```

```
tdf["top_first"].value_counts().plot(kind="barh", figsize=(15,6))
```

<Axes: >




```
## Top 2 positions are backend engineer and fullstack engineer
```

```
## Top 10 employees in Amazon- X department - having 5/6/7 years of experience earning more than their peers - Tier X
```

```
df2[(df2["Class"]==1) & (df2["TYOE"]>=5)].head(10)
```

	company_hash	email_hash	orgyear	ctc	job_position	ctc_updated_year	TYOE	Designat
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	1100000	other	2020.0	7.0	
4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	2017.0	1400000	fullstack engineer	2019.0	6.0	
11	ngdor ntwy	72c2171a022115d475c8faac306912a4c95f6dd7fdd320...	2016.0	600000	ios engineer	2021.0	7.0	
15	bgsrxd	3b99c28818530737364245236fba9a821187fc38cd6445...	2012.0	2030000	backend engineer	2019.0	11.0	
27	wvuxrrvqj ntwyzgrgsxto	86b90fa2f295d246b8cef7858209c9427ac09b0074a7aa...	2012.0	2200000	frontend engineer	2019.0	11.0	
30	qxenxg	65801e6e2e0d70deafcd5fcd0b476af33759c79906f692...	2014.0	2600000	backend engineer	2019.0	9.0	
32	gvnx	7ed9dad40408750d848b8c1e568746be7ac2947ec098e6...	2013.0	780000	frontend engineer	2021.0	10.0	
34	bxzanqtt	72778e5ee3cd195927e924462a22c2e736920541766327...	2011.0	1500000	android engineer	2021.0	12.0	
35	qtwpgzojo ntwy rvmo ucn rna	ba5454243306a2afe8da0731ac189480d2c70fcf694417...	2015.0	1500000	fullstack engineer	2019.0	8.0	
43	ogqgwg	738318c479954b0dc59d17cf391a9fc9a4b0a5ab42aaf4...	2011.0	2500000	frontend engineer	2019.0	12.0	

```
df2.head()
```

	company_hash	email_hash	orgyear	ctc	job_position	ctc_updated_year	TYOE	Designatic
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	1100000	other	2020.0	7.0	
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018.0	449999	fullstack engineer	2019.0	5.0	
2	vx	[illegible]	[illegible]	[illegible]	engineer	[illegible]	[illegible]	
4	gxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520	2017.0	1400000	fullstack	2019.0	6.0	

```
# (E) Data processing for Unsupervised clustering - Label encoding/ One- hot encoding, Standardization of data
```

```
df2.shape
```

    (144588, 10)

```
df2["job_position"].nunique()
```

    36

```
X = df2.loc[:,["ctc","job_position","TYOE","Designation","Class","Tier"]]
```

```
X.head()
```

	ctc	job_position	TYOE	Designation	Class	Tier
0	1100000	other	7.0	1	1	3
1	449999	fullstack engineer	5.0	3	3	3
2	2000000	backend engineer	8.0	1	2	2
3	700000	backend engineer	6.0	3	3	3
4	1400000	fullstack engineer	6.0	1	1	3

```
X.shape
```

    (144588, 6)

```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import OneHotEncoder
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
```

```
X_encoded = X.copy()
```

```
X_encoded["job_position"]=X_encoded["job_position"].map(dict(df2["job_position"].value_counts()))
```

```
X_encoded.isnull().sum()
```

```
ctc             0
job_position    0
TYOE            0
Designation     0
Class           0
Tier            0
dtype: int64
```

```
scaler = MinMaxScaler()
```

```
scaler.fit(X_encoded)
```

```
▾ MinMaxScaler
MinMaxScaler()
```

```
X_scaled = scaler.transform(X_encoded)
```

```
pd.DataFrame(X_scaled, columns=X.columns).head()
```

	ctc	job_position	TYOE	Designation	Class	Tier
0	0.185753	0.432115	0.098039	0.0	0.0	1.0
1	0.064917	0.593979	0.058824	1.0	1.0	1.0

```
## clustering tendency: pairplot
```

```
sns.pairplot(pd.DataFrame(X_scaled, columns=X.columns))
```

<seaborn.axisgrid.PairGrid at 0x7ed3d4cd5810>


```
# KMeans clustering
```

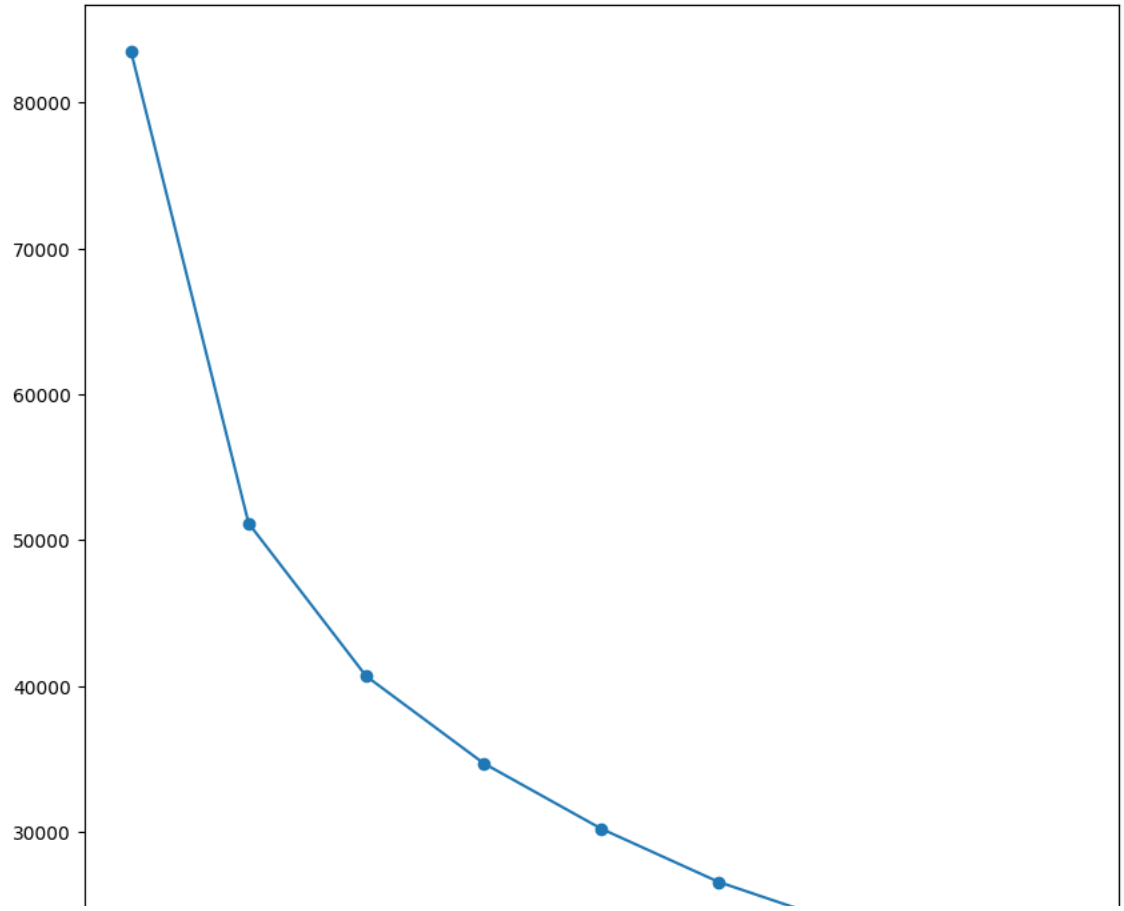
```
# clustering tendency: Elbow method
```

```
from sklearn.cluster import KMeans
```

```
wcss = []
for k in range(1, 10):
    model = KMeans(n_clusters = k)
    model.fit(X_scaled)
    wcss.append(model.inertia_)
```

```
plt.plot(range(1, 10), wcss, '-o')
```

```
[<matplotlib.lines.Line2D at 0x7ed3d2902920>]
```




```
## k=3 possible clusters
```

```
kmeans = KMeans(n_clusters=3)
kmeans.fit(X_scaled)
```

```
▾ KMeans
KMeans(n_clusters=3)
```

```
kmeans.predict(X_scaled)
```

```
array([0, 2, 0, ..., 1, 1, 0], dtype=int32)
```

```
kmeans.labels_
```

```
array([0, 2, 0, ..., 1, 1, 0], dtype=int32)
```

```
clusters = X.copy()
```

```
clusters["label"]=kmeans.labels_
```

```
clusters.head()
```

	ctc	job_position	TYOE	Designation	Class	Tier	label
0	1100000	other	7.0	1	1	3	0
1	449999	fullstack engineer	5.0	3	3	3	2
2	2000000	backend engineer	8.0	1	2	2	0
3	700000	backend engineer	6.0	3	3	3	2
4	1400000	fullstack engineer	6.0	1	1	3	0

```
## description of cluster 0
```

```
clusters[clusters["label"]==0][["ctc","TYOE","Designation","Class","Tier"]].describe()
```

	ctc	TYOE	Designation	Class	Tier
count	4.185200e+04	41852.000000	41852.000000	41852.000000	41852.000000
mean	2.419096e+06	9.331884	1.200779	1.654497	1.769856
std	9.629076e+05	4.882936	0.420607	0.651468	0.629518
min	4.700000e+05	2.000000	1.000000	1.000000	1.000000
25%	1.700000e+06	6.000000	1.000000	1.000000	1.000000
50%	2.200000e+06	8.000000	1.000000	2.000000	2.000000
75%	3.000000e+06	12.000000	1.000000	2.000000	2.000000
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

```
#description of cluster 1
```

```
clusters[clusters["label"]==1][["ctc","TYOE","Designation","Class","Tier"]].describe()
```

	ctc	TYOE	Designation	Class	Tier
count	6.572600e+04	65726.000000	65726.000000	65726.000000	65726.000000
mean	7.692233e+05	7.982275	2.660378	2.212153	2.944969
std	4.277847e+05	4.030621	0.504690	0.613822	0.237649
min	1.008000e+05	2.000000	1.000000	1.000000	1.000000
25%	4.500000e+05	5.000000	2.000000	2.000000	3.000000
50%	7.000000e+05	7.000000	3.000000	2.000000	3.000000
75%	1.000000e+06	10.000000	3.000000	3.000000	3.000000
max	3.500000e+06	53.000000	3.000000	3.000000	3.000000

```
#description of cluster 2
```

```
clusters[clusters["label"]==2][["ctc","TYOE","Designation","Class","Tier"]].describe()
```

	ctc	TYOE	Designation	Class	Tier
count	3.701000e+04	37010.000000	37010.000000	37010.000000	37010.000000
mean	8.971730e+05	7.224156	2.674926	2.533531	2.875196
std	4.559644e+05	3.203149	0.484738	0.623360	0.330583
min	1.010000e+05	2.000000	1.000000	1.000000	1.000000
25%	5.000000e+05	5.000000	2.000000	2.000000	3.000000
50%	8.300000e+05	7.000000	3.000000	3.000000	3.000000
75%	1.200000e+06	9.000000	3.000000	3.000000	3.000000
max	2.850000e+06	40.000000	3.000000	3.000000	3.000000

```
clusters["label"].value_counts(normalize=True)
```

```
1    0.454574
0    0.289457
2    0.255969
Name: label, dtype: float64
```

```
## Hierarchical clusteringg
```

```
col_agg = {"ctc":"median","job_position":"max","TYOE":"mean","Designation":"mean","Class":"mean","Tier":"mean"}
```

```
df_new = df2.groupby(["job_position"]).aggregate(col_agg); df_new.head()
```

ctc	job_position	TYOE	Designation	Class	Tier

```
df_new["job_position"]=df_new["job_position"].map(dict(df["job_position"].value_counts()))
```

android engineer	900000.0	android engineer	8.825381	2.245405	2.060227	2.714314

```
df_new.head()
```

	ctc	job_position	TYOE	Designation	Class	Tier
job_position						
android engineer	900000.0	NaN	8.825381	2.245405	2.060227	2.714314
backend architect	2600000.0	NaN	14.493945	2.227509	2.077855	1.706747
backend engineer	1200000.0	NaN	7.797302	2.237725	2.176203	2.489572
cofounder	1200000.0	NaN	9.462963	2.206790	2.018519	2.379630
consultant	1200000.0	1.0	9.803922	2.117647	2.078431	2.490196

```
minmax_scale = MinMaxScaler()
```

```
minmax_scale.fit(df_new)
```

```
▾ MinMaxScaler
MinMaxScaler()
```

```
X_new = pd.DataFrame(minmax_scale.transform(df_new), columns=df_new.columns, index=df_new.index)
```

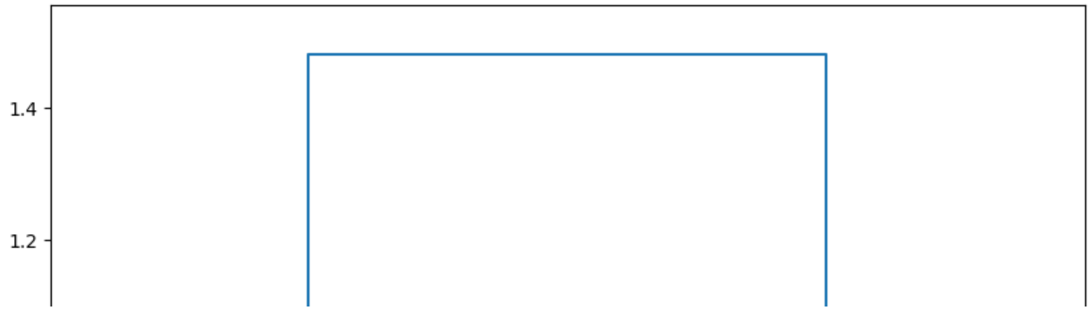
```
X_new.isnull().sum()
```

```
ctc             0
job_position   32
TYOE            0
Designation     0
Class           0
Tier            0
dtype: int64
```

```
X_new = X_new.dropna(how='any')
```

```
from scipy.cluster.hierarchy import dendrogram, linkage
linkage_data = linkage(X_new, method='ward', metric='euclidean')
dendrogram(linkage_data)

plt.show()
```

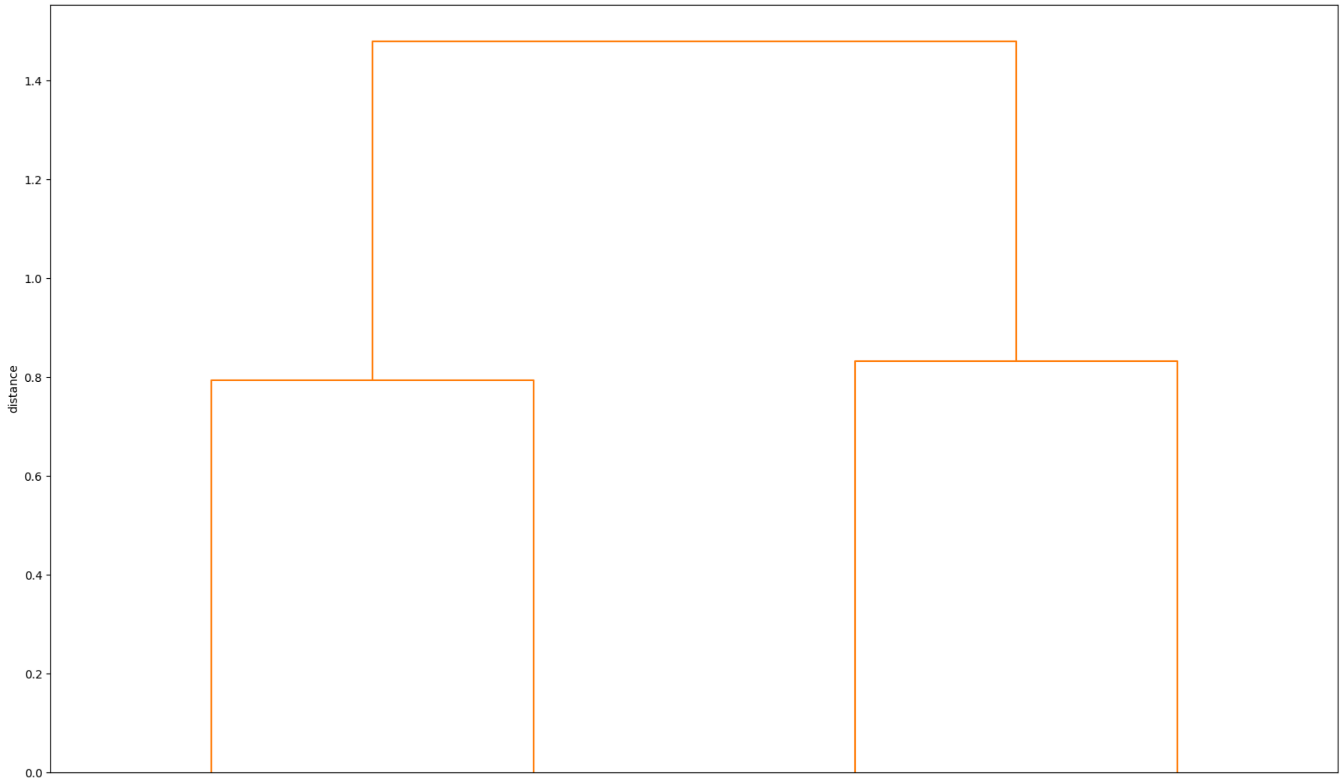



```
import scipy.cluster.hierarchy as sch

Z = sch.linkage(X_new, method='ward', metric='euclidean')

fig, ax = plt.subplots(figsize=(20, 12))
sch.dendrogram(Z, labels=X_new.index, ax=ax, color_threshold=2)
plt.xticks(rotation=90)
ax.set_ylabel('distance')
```

```
Text(0, 0.5, 'distance')
```




```
clusters["label"].value_counts(normalize=True)
```

```
1    0.454574
0    0.289457
2    0.255969
Name: label, dtype: float64
```

```
# observations from Kmeans clustering:

# 41% people with median 4 years of experience are in cluster 1
# 28% people with median 5 years of experience are in cluster 2
# 31% people with median 3 years of experience are in cluster 0


# approx 9.9% companies are Tier 1
# approx 21.45% companies are Tier 2
# approx 68.57% companies are Tier 3
# Top 2 positions are backend engineer and fullstack engineer

# Recommendations:
# Since 68.57% companies are Tier3, employees from these Tier3 companies will have higher chances to
# enroll for scaler programmes  and change their domain from non-IT to IT.
# Employees from Tier2 companies will mostly enroll for skill up in their current domain.
```