Scaler - Clustering

Defining Problem Statement & Exploratory Data Analysis

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

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style = 'darkgrid')
import datetime as dt
import re

import warnings # supress warnings
warnings.filterwarnings('ignore')
```

Importing Data & removing non-relevant columns / duplicates

In [2]:
```
#importing data
raw_data = pd.read_csv('scaler_clustering.csv')
```

In [3]:
```
raw_data.shape
```

Out[3]: (205843, 7)

In [4]:
```
#creating copy of imported data and removing spaces from column names
data = raw_data.copy(deep = True)
data.columns = data.columns.str.strip()
data.columns = data.columns.str.replace(' ', '_')
data.columns = data.columns.str.replace('-', '_')
data.sample(100).head()
```

Out[4]:

	Unnamed:_0	company_hash	email_hash	orgyear	ctc	job_position	ctc_updated_year
50057	50113	grv vzx ntwyzgrgsxto ucn rna	11429f41e35ce9e6f541ee504c64469a45645a06a746a4...	2013.0	1900000	Frontend Engineer	2017.0
15948	15957	vrnqvz ntwyzgrgsxto	e2d83c8ef1f08126a1ad4b2da561aacba850f6f5a77545...	2014.0	500000	Other	2021.0
183456	184357	dtmxv xn	bf2f6843105f06c50c20eb03dbab5c325c85b33c5be37a...	2010.0	2250000	NaN	2020.0
78289	78402	egqa bgngq wgbuvzj	923f1c306fcd1af3d2f1ca493e59569a5f5cdbb83292c0...	2015.0	730000	FullStack Engineer	2017.0
146674	147226	srgmvravnv	7ac8cb8d8d902327159c972c530a2ed35c1bde4d0eab9e...	2015.0	900000	Backend Engineer	2020.0

In [5]:
```
#dropping non-relevant columns
data.drop(columns = ['Unnamed:_0'],axis = 1, inplace = True)
#data.drop(columns = ['email_hash'],axis = 1, inplace = True)

data.drop_duplicates(keep='last', inplace = True)
data.shape
```

Out[5]: (205810, 6)

Data pre-processing & Treating Null values in various columns

In [6]:
```
def preprocess_string(string):
    new_string= re.sub('[^A-Za-z ]+', '', string).lower().strip()
    return new_string

#pre-processing Job Positions column using regex
print(f'Unique Job Positions before preprocessing: {data.job_position.nunique()}')
data.job_position=data.job_position.apply(lambda x: preprocess_string(str(x)))
data.drop_duplicates(keep='last', inplace = True)
print(f'Unique Job Positions after preprocessing: {data.job_position.nunique()}')
```

```
Unique Job Positions before preprocessing: 1017
Unique Job Positions after preprocessing: 857
```

In [7]:
```
#pre-processing company_hash column using regex
print(f'Unique company_hash before preprocessing: {data.company_hash.nunique()}')
data.company_hash=data.company_hash.apply(lambda x: preprocess_string(str(x)))
data.drop_duplicates(keep='last', inplace = True)
print(f'Unique company_hash after preprocessing: {data.company_hash.nunique()}')
```

```
Unique company_hash before preprocessing: 37299
Unique company_hash after preprocessing: 37208
```

In [8]:
```
#removing rows where company or job_position is not available
data=data[ ~((data['company_hash']=='') | (data['job_position']==''))]
data.shape
```

Out[8]: (205603, 6)

In [9]:
```
data.isnull().sum()
```

Out[9]:
```
company_hash         0
email_hash           0
orgyear             86
ctc                  0
job_position         0
ctc_updated_year     0
dtype: int64
```

In [10]:
```
## Since the count of rows is small, I will drop the column
## we could have used Knn Immutation of mean imputation, still based on understanding, it is better to drop these rows
```

```
data.dropna(inplace = True)
data.shape
```

Out[10]:
```
(205517, 6)
```

In [11]:
```
data.isnull().sum()
```

Out[11]:
```
company_hash        0
email_hash          0
orgyear             0
ctc                 0
job_position        0
ctc_updated_year    0
dtype: int64
```

Univariate Analysis & Removing Outliers

Numerical Columns

In [12]:
```
data.describe(include = np.number, percentiles=[.25,.5,.75,.90,.95, .99, .999]).round(2).T
```

Out[12]:

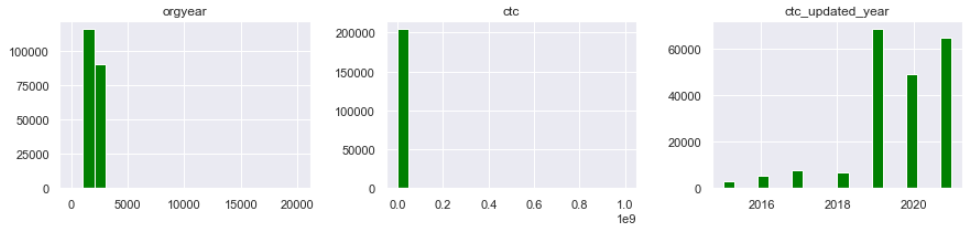
	count	mean	std	min	25%	50%	75%	90%	95%	99%	99.9%	max
orgyear	205517.0	2014.88	63.61	0.0	2013.0	2016.0	2018.0	2019.0	2020.0	2021.0	2023.0	2.016500e+04
ctc	205517.0	2270525.17	11794185.93	2.0	530000.0	950000.0	1700000.0	2800000.0	3800000.0	12600000.0	200000000.0	1.000150e+09
ctc_updated_year	205517.0	2019.63	1.33	2015.0	2019.0	2020.0	2021.0	2021.0	2021.0	2021.0	2021.0	2.021000e+03

In [13]:
```
data.describe(include = 'object').round(2).T
```

Out[13]:

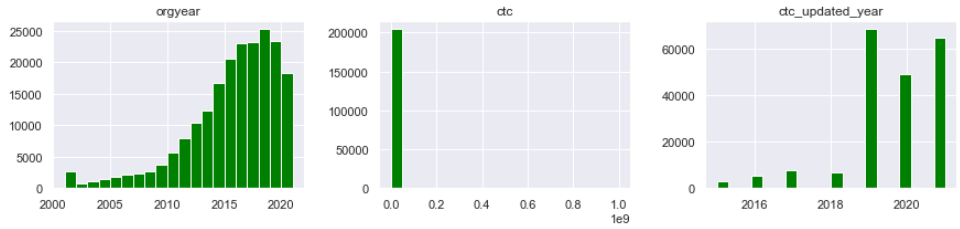
	count	unique	top	freq
company_hash	205517	37180	nvnv wgzohrnvzwj otqcxwto	8335
email_hash	205517	153303	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	10
job_position	205517	856	nan	52487

In [14]:
```
#Creating Histograms for Continuous Variables
data.hist(figsize=(20,10), bins=20, layout=(3,4), color ='green')
plt.show()
```



In [15]:
```
#Clipping Column orgyear & ctc
data.orgyear = data.orgyear.clip(data.orgyear.quantile(0.01), data.orgyear.quantile(0.99))
#data.ctc = data.ctc.clip(data.ctc.quantile(0.01), data.ctc.quantile(0.99))

data['orgyear'] = data['orgyear'].astype('int')
data['ctc_updated_year'] = data['ctc_updated_year'].astype('int')
```

In [16]:
```
#Creating Histograms for Continuous Variables
data.hist(figsize=(20,10), bins=20, layout=(3,4), color ='green')
plt.show()
```



Categorical Columns

In [17]:
```
#Masking companies by renaming it to 'Others' having count less than or equal to 5
data.company_hash.value_counts()
```

Out[17]:
```
nvnv wgzohrnvzwj otqcxwto    8335
xzegojo                      5381
vbvkgz                       3480
zgn vuurxwvmrt vwwghzn       3410
wgszxkvzn                    3238
                             ...
otwhqtrvjtq                     1
wxnxktzo vrrxvzwt ucn rna       1
ltzgwqvtnxcto ucn rna           1
pqvnxpvr ntwy ucn rna           1
bvptbjnqxu td vbvkgz            1
Name: company_hash, Length: 37180, dtype: int64
```

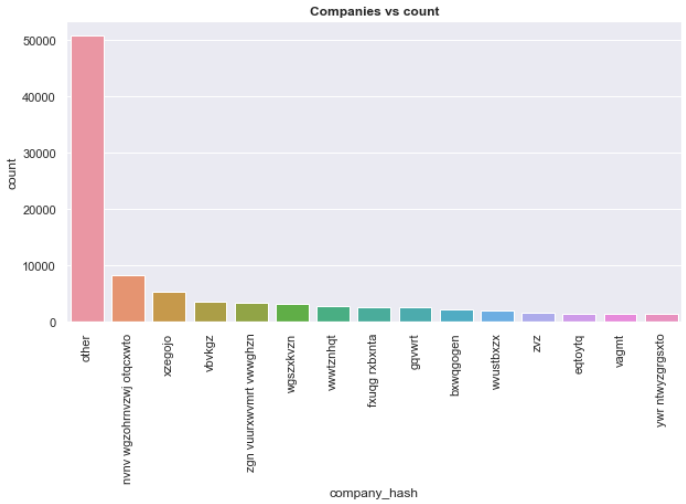
In [18]:
```
data.loc[data.groupby('company_hash')['ctc'].transform('count') <= 5, 'company_hash'] = 'other'
data['company_hash'] = data['company_hash'].fillna('other')
```

In [19]:
```
data.company_hash.value_counts()
```

Out[19]:
```
other                        50788
nvnv wgzohrnvzwj otqcxwto     8335
xzegojo                       5381
vbvkgz                        3480
zgn vuurxwvmrt vwwghzn        3410
                             ...
oow vactzn                       6
yhqgz                            6
uvqvsgz axsxnvr otqcxwto         6
svbtwyvzst ogrhnxgzo             6
```

```
ohbngnvr ojontbo                6
Name: company_hash, Length: 3169, dtype: int64
```

In [20]:
```
#Countplot for company_hash
x = data['company_hash'].value_counts().sort_values(ascending=False).head(15).index
y = data['company_hash'].value_counts().sort_values(ascending=False).head(15).values
plt.figure(figsize=(10,5))
sns.barplot(x=x, y=y)
plt.xlabel('company_hash')
plt.ylabel('count')
plt.xticks(rotation=90)
plt.title('Companies vs count', fontsize = 12, fontweight = 'bold')
plt.show()
```



In [21]:
```
#Masking job_positions by renaming it to 'Others' having count less than or equal to 5
data.job_position.value_counts()
```

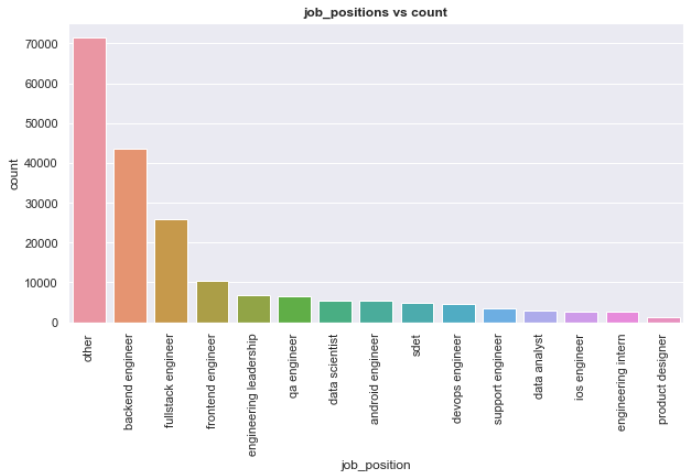
Out[21]:
```
nan                                52487
backend engineer                   43520
fullstack engineer                 25863
other                              18050
frontend engineer                  10409
                                   ...
senior engineer software               1
software enginnering specialist        1
android lead                           1
senior analysts                        1
azure data factory                     1
Name: job_position, Length: 856, dtype: int64
```

In [22]:
```
#Filling null values with others  -- if not done before
data.loc[data.groupby('job_position')['ctc'].transform('count') <= 5, 'job_position'] = 'other'
data.loc[data['job_position']=='nan', 'job_position']=np.nan
data['job_position'] = data['job_position'].fillna('other')
```

In [23]:
```
data.job_position.value_counts().head(20)
```

Out[23]:
```
other                     71562
backend engineer          43520
fullstack engineer        25863
frontend engineer         10409
engineering leadership     6864
qa engineer                6584
data scientist             5367
android engineer           5346
sdet                       4971
devops engineer            4609
support engineer           3599
data analyst               2902
ios engineer               2743
engineering intern         2691
product designer           1313
backend architect          1287
research engineers         1228
product manager            1161
program manager             814
non coder                   595
Name: job_position, dtype: int64
```

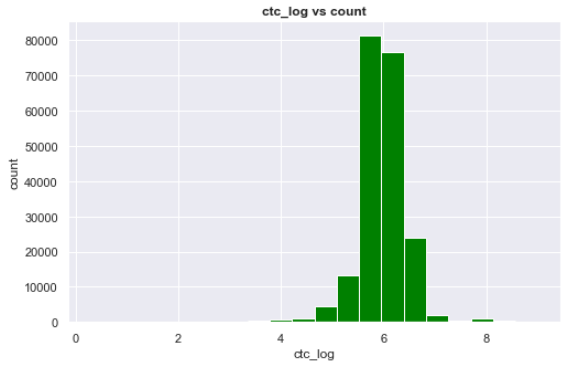
In [24]:
```
#Countplot for job_position
x = data['job_position'].value_counts().sort_values(ascending=False).head(15).index
y = data['job_position'].value_counts().sort_values(ascending=False).head(15).values
plt.figure(figsize=(10,5))
sns.barplot(x=x, y=y)
plt.xlabel('job_position')
plt.ylabel('count')
plt.xticks(rotation=90)
plt.title('job_positions vs count', fontsize = 12, fontweight = 'bold')
plt.show()
```



Feature Engineering : Creating new columns & Droping old columns

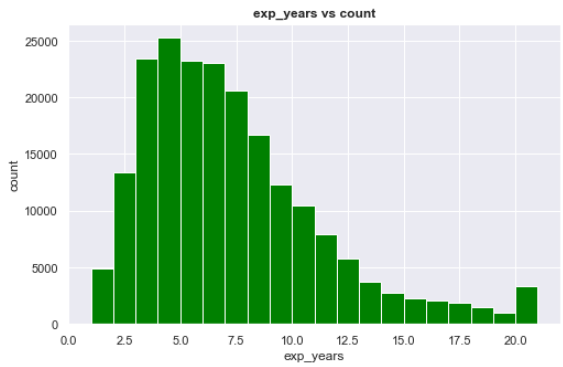
In [25]:
```
#creating column - Log (ctc)
data['ctc_log'] = np.log10(data['ctc'])
```

In [26]:
```
data['ctc_log'].hist(figsize=(8,5), bins=20, color ='green')
plt.xlabel('ctc_log')
plt.ylabel('count')
plt.title('ctc_log vs count', fontsize = 12, fontweight = 'bold')
plt.show()
```



In [27]:
```
#create new column - exp_years
#Making new feature like adding 'Years of Experience' column by subtracting orgyear from current year
data['exp_years'] = data['orgyear'].max()+1-data['orgyear']
data=data[~data['exp_years'].isnull()]
```

In [28]:
```
data['exp_years'].hist(figsize=(8,5), bins=20, color ='green')
plt.xlabel('exp_years')
plt.ylabel('count')
plt.title('exp_years vs count', fontsize = 12, fontweight = 'bold')
plt.show()
```



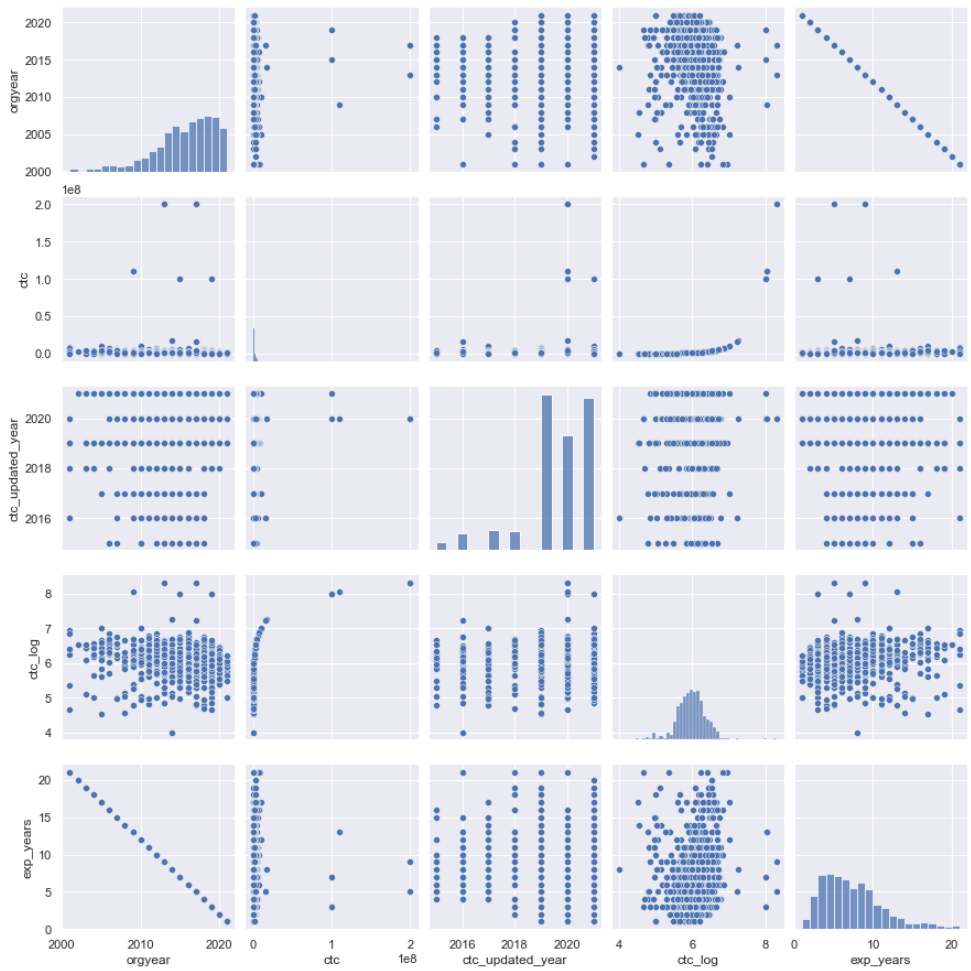
In [29]:
```
data.describe(include = np.number, percentiles=[.25,.5,.75,.90,.95, .99, .999]).round(2).T
```

Out[29]:

	count	mean	std	min	25%	50%	75%	90%	95%	99%	99.9%	max
orgyear	205517.0	2015.14	4.06	2001.0	2013.00	2016.00	2018.00	2019.00	2020.00	2021.0	2021.0	2.021000e+03
ctc	205517.0	2270525.17	11794185.93	2.0	530000.00	950000.00	1700000.00	2800000.00	3800000.00	12600000.0	200000000.0	1.000150e+09
ctc_updated_year	205517.0	2019.63	1.33	2015.0	2019.00	2020.00	2021.00	2021.00	2021.00	2021.0	2021.0	2.021000e+03
ctc_log	205517.0	5.97	0.46	0.3	5.72	5.98	6.23	6.45	6.58	7.1	8.3	9.000000e+00
exp_years	205517.0	6.86	4.06	1.0	4.00	6.00	9.00	12.00	15.00	21.0	21.0	2.100000e+01

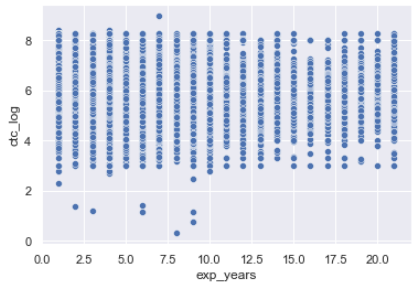
Bivariate Analysis

In [30]:
```
#Using PairPlot to plot scatter plots for all columns
sns.pairplot(data.sample(1000))
plt.show()
```



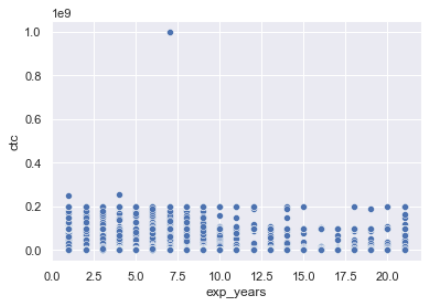
In [31]:
```
#plotting scatter plot between years of experience vs CTC_log
sns.scatterplot(data= data, x = 'exp_years', y = 'ctc_log' )
```

Out[31]:
```
<AxesSubplot:xlabel='exp_years', ylabel='ctc_log'>
```



In [32]:
```
#plotting scatter plot between years of experience vs CTC
sns.scatterplot(data= data, x = 'exp_years', y = 'ctc' )
```

Out[32]:
```
<AxesSubplot:xlabel='exp_years', ylabel='ctc'>
```



In [33]:
```
data.head(3)
```

Out[33]:

	company_hash	email_hash	orgyear	ctc	job_position	ctc_updated_year	ctc_log	exp_years
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016	1100000	other	2020	6.041393	6
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018	449999	fullstack engineer	2019	5.653212	4
2	other	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2015	2000000	backend engineer	2020	6.301030	7

In [34]:
```
data2 = data.drop(columns = ['orgyear', 'ctc_updated_year'], axis = 1)
```

In [35]:
```
data2.head()
```

Out[35]:

	company_hash	email_hash	ctc	job_position	ctc_log	exp_years
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	1100000	other	6.041393	6
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	449999	fullstack engineer	5.653212	4

	company_hash	email_hash	ctc	job_position	ctc_log	exp_years
2	other	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2000000	backend engineer	6.301030	7
3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	700000	backend engineer	5.845098	5
4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	1400000	fullstack engineer	6.146128	5

Manual Clustering of Data :

1. Based on Company, Job position & Years of experience : designation

In [36]:
```
def segment(a, b_50, b_75):
    if a >= b_75: return 1
    elif a >= b_50: return 2
    else: return 3
```

In [37]:
```
group_c_jp_y = data2.groupby(['company_hash','job_position','exp_years'])['ctc_log'].describe()
group_c_jp_y
```

Out[37]:

			count	mean	std	min	25%	50%	75%	max
company_hash	**job_position**	**exp_years**								
a ntwyzgrgsxto	**android engineer**	**8**	1.0	5.851258	NaN	5.851258	5.851258	5.851258	5.851258	5.851258
	backend engineer	**7**	1.0	5.778151	NaN	5.778151	5.778151	5.778151	5.778151	5.778151
		8	1.0	5.778151	NaN	5.778151	5.778151	5.778151	5.778151	5.778151
		9	1.0	6.130334	NaN	6.130334	6.130334	6.130334	6.130334	6.130334
	frontend engineer	**6**	2.0	5.698970	0.000000	5.698970	5.698970	5.698970	5.698970	5.698970
...	**...**	**...**	...	...	...	...	...	...	...	...
zxztrtvuo	**other**	**3**	8.0	5.832326	0.368323	5.602060	5.640424	5.676091	5.798455	6.653213
		4	2.0	6.001730	0.221511	5.845098	5.923414	6.001730	6.080046	6.158362
		6	2.0	6.079181	0.000000	6.079181	6.079181	6.079181	6.079181	6.079181
		7	2.0	6.238561	0.225396	6.079181	6.158871	6.238561	6.318250	6.397940
		8	1.0	6.105510	NaN	6.105510	6.105510	6.105510	6.105510	6.105510

59513 rows × 8 columns

In [38]:
```
data_c_jp_y = data2.merge(group_c_jp_y, how = 'left', on = ['company_hash','job_position','exp_years'])
```

In [39]:
```
data_c_jp_y.head(2)
```

Out[39]:

	company_hash	email_hash	ctc	job_position	ctc_log	exp_years	count	mean	std	min	25%	50%	75%	max
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	1100000	other	6.041393	6	1.0	6.041393	NaN	6.041393	6.041393	6.041393	6.041393	6.041393
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	449999	fullstack engineer	5.653212	4	7.0	5.869306	0.14164	5.653212	5.785271	5.875061	5.953571	6.079181

In [40]:
```
data_c_jp_y['designation'] =data_c_jp_y.apply(lambda x: segment(x['ctc_log'],x['50%'],x['75%']),axis=1)
```

In [41]:
```
data_c_jp_y.head()
```

Out[41]:

	company_hash	email_hash	ctc	job_position	ctc_log	exp_years	count	mean	std	min	25%	50%	75%	max	designation
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	1100000	other	6.041393	6	1.0	6.041393	NaN	6.041393	6.041393	6.041393	6.041393	6.041393	1
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	449999	fullstack engineer	5.653212	4	7.0	5.869306	0.141640	5.653212	5.785271	5.875061	5.953571	6.079181	3
2	other	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2000000	backend engineer	6.301030	7	923.0	5.879407	0.508124	3.000000	5.698970	5.954243	6.176091	8.001820	1
3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	700000	backend engineer	5.845098	5	7.0	6.040962	0.153013	5.845098	5.916254	6.079181	6.143453	6.243038	3
4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	1400000	fullstack engineer	6.146128	5	1.0	6.146128	NaN	6.146128	6.146128	6.146128	6.146128	6.146128	1

In [42]:
```
data3 = data_c_jp_y.drop(columns = ['count', 'mean', 'std','min', '25%','50%', '75%', 'max'], axis = 1)
```

In [43]:
```
data3.head()
```

Out[43]:

	company_hash	email_hash	ctc	job_position	ctc_log	exp_years	designation
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	1100000	other	6.041393	6	1
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	449999	fullstack engineer	5.653212	4	3
2	other	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2000000	backend engineer	6.301030	7	1
3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	700000	backend engineer	5.845098	5	3
4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	1400000	fullstack engineer	6.146128	5	1

1. Based on Company, Job position : classs

In [44]:
```
group_c_jp = data3.groupby(['company_hash','job_position'])['ctc_log'].describe()
group_c_jp
```

Out[44]:

		count	mean	std	min	25%	50%	75%	max
company_hash	**job_position**								
a ntwyzgrgsxto	**android engineer**	1.0	5.851258	NaN	5.851258	5.851258	5.851258	5.851258	5.851258
	backend engineer	3.0	5.895545	0.203333	5.778151	5.778151	5.778151	5.954243	6.130334
	frontend engineer	2.0	5.698970	0.000000	5.698970	5.698970	5.698970	5.698970	5.698970
	fullstack engineer	2.0	5.668230	0.175591	5.544068	5.606149	5.668230	5.730311	5.792392
	other	8.0	6.076703	0.438936	5.544068	5.744493	5.954243	6.524248	6.602060
...	**...**	...	...	...	...	...	...	...	...

company_hash	job_position	count	mean	std	min	25%	50%	75%	max
zxztrtvuo	engineering intern	3.0	5.751758	0.181929	5.602060	5.650515	5.698970	5.826606	5.954243
	frontend engineer	16.0	5.954128	0.269996	5.653213	5.740363	5.879456	6.123204	6.397940
	fullstack engineer	7.0	5.910781	0.171885	5.698970	5.802383	5.851258	6.022191	6.176091
	ios engineer	1.0	6.079181	NaN	6.079181	6.079181	6.079181	6.079181	6.079181
	other	25.0	5.871804	0.277484	5.602060	5.653213	5.740363	6.079181	6.653213

21254 rows × 8 columns

In [45]:
```
data_c_jp = data3.merge(group_c_jp, how = 'left', on = ['company_hash','job_position'])
```

In [46]:
```
data_c_jp.head(2)
```

Out[46]:

	company_hash	email_hash	ctc	job_position	ctc_log	exp_years	designation	count	mean	std	min	25%	50%	75%	max
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	1100000	other	6.041393	6	1	2.0	6.035388	0.008492	6.029384	6.032386	6.035388	6.038390	6.041393
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	449999	fullstack engineer	5.653212	4	3	25.0	5.940805	0.226093	5.477121	5.778151	5.929419	6.139879	6.301030

In [47]:
```
data_c_jp['classs'] =data_c_jp.apply(lambda x: segment(x['ctc_log'],x['50%'],x['75%']),axis=1)
```

In [48]:
```
data_c_jp.head()
```

Out[48]:

	company_hash	email_hash	ctc	job_position	ctc_log	exp_years	designation	count	mean	std	min	25%	50%	75%	max	classs
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	1100000	other	6.041393	6	1	2.0	6.035388	0.008492	6.029384	6.032386	6.035388	6.038390	6.041393	1
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	449999	fullstack engineer	5.653212	4	3	25.0	5.940805	0.226093	5.477121	5.778151	5.929419	6.139879	6.301030	3
2	other	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2000000	backend engineer	6.301030	7	1	7989.0	5.872645	0.550579	3.000000	5.653213	5.954243	6.176091	8.301030	1
3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	700000	backend engineer	5.845098	5	3	25.0	6.134792	0.195517	5.716003	6.021189	6.187521	6.255273	6.544068	3
4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	1400000	fullstack engineer	6.146128	5	1	3.0	5.885558	0.226817	5.732394	5.755273	5.778151	5.962140	6.146128	1

In [49]:
```
data4 = data_c_jp.drop(columns = ['count', 'mean', 'std','min', '25%','50%', '75%', 'max'], axis = 1)
```

In [50]:
```
data4.head()
```

Out[50]:

	company_hash	email_hash	ctc	job_position	ctc_log	exp_years	designation	classs
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	1100000	other	6.041393	6	1	1
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	449999	fullstack engineer	5.653212	4	3	3
2	other	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2000000	backend engineer	6.301030	7	1	1
3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	700000	backend engineer	5.845098	5	3	3
4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	1400000	fullstack engineer	6.146128	5	1	1

1. Based on Company, Job position : tier

In [51]:
```
group_c = data4.groupby(['company_hash'])['ctc_log'].describe()
group_c
```

Out[51]:

company_hash	count	mean	std	min	25%	50%	75%	max
a ntwyzgrgsxto	16.0	5.930370	0.354768	5.544068	5.698970	5.778151	6.130334	6.602060
aaqxctz avnv owxtzwto vzvrjnxwo ucn rna	8.0	5.841436	0.343894	5.556303	5.663303	5.698970	5.901190	6.556303
adw ntwyzgrgsj	300.0	5.822969	0.366316	4.176091	5.602060	5.778151	6.060698	8.000000
adw ntwyzgrgsxto	140.0	5.862755	0.394856	5.000000	5.623249	5.835586	6.000000	8.000000
agdutq	6.0	6.189559	0.186852	5.903090	6.146128	6.146128	6.334987	6.397940
...	...	...	...	...	...	...	...	...
zxxn rna	7.0	5.883398	0.281066	5.447158	5.698970	5.908485	6.099878	6.230449
zxyxrtzn	8.0	5.795894	0.206868	5.518514	5.621665	5.851215	5.909672	6.113943
zxyxrtzn ntwyzgrgsxto	12.0	5.818693	0.385525	4.698970	5.770660	5.903090	5.994333	6.176091
zxzlvwvqn	42.0	6.137596	0.308149	5.255273	5.903090	6.130036	6.337372	6.698970
zxztrtvuo	77.0	5.947010	0.283610	5.602060	5.698970	5.913813	6.096910	7.077368

3169 rows × 8 columns

In [52]:
```
data_c = data4.merge(group_c, how = 'left', on = ['company_hash'])
```

In [53]:
```
data_c.head(2)
```

Out[53]:

	company_hash	email_hash	ctc	job_position	ctc_log	exp_years	designation	classs	count	mean	std	min	25%	50%	75%	max
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	1100000	other	6.041393	6	1	1	9.0	6.011750	0.192448	5.69897	5.903090	6.029384	6.176091	6.248219
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	449999	fullstack engineer	5.653212	4	3	3	428.0	5.983985	0.382884	4.00000	5.778151	5.954243	6.227562	8.301030

In [54]:
```
data_c['tier'] =data_c.apply(lambda x: segment(x['ctc_log'],x['50%'],x['75%']),axis=1)
```

In [55]:
```
data_c.head()
```

Out[55]:

	company_hash	email_hash	ctc	job_position	ctc_log	exp_years	designation	classs	count	mean	std	min	25%	50%	75%	max	tier

	company_hash	email_hash	ctc	job_position	ctc_log	exp_years	designation	classs	count	mean	std	min	25%	50%	75%	max	tier
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	1100000	other	6.041393	6	1	1	9.0	6.011750	0.192448	5.698970	5.903090	6.029384	6.176091	6.248219	2
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	449999	fullstack engineer	5.653212	4	3	3	428.0	5.983985	0.382884	4.000000	5.778151	5.954243	6.227562	8.301030	3
2	other	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2000000	backend engineer	6.301030	7	1	1	50788.0	5.875194	0.521964	1.176091	5.643453	5.903090	6.146128	9.000065	1
3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	700000	backend engineer	5.845098	5	3	3	70.0	6.166785	0.252216	5.301030	6.041393	6.146128	6.301030	6.672098	3
4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	1400000	fullstack engineer	6.146128	5	1	1	6.0	5.941317	0.182685	5.732394	5.794888	5.922549	6.109596	6.146128	1

In [56]:
```
data5 = data_c.drop(columns = ['count', 'mean', 'std','min', '25%','50%', '75%', 'max'], axis = 1)
```

In [57]:
```
data5.head()
```

Out[57]:

	company_hash	email_hash	ctc	job_position	ctc_log	exp_years	designation	classs	tier
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	1100000	other	6.041393	6	1	1	2
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	449999	fullstack engineer	5.653212	4	3	3	3
2	other	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2000000	backend engineer	6.301030	7	1	1	1
3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	700000	backend engineer	5.845098	5	3	3	3
4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	1400000	fullstack engineer	6.146128	5	1	1	1

Questions based on Manual Clustering.

Q1. Top 10 employees (earning more than most of the employees in the company) - Tier 1

In [58]:
```
q1 = data5.loc[data5['tier'] == 1].sort_values(by=['ctc_log'], ascending=False).head(10)
q1
```

Out[58]:

	company_hash	email_hash	ctc	job_position	ctc_log	exp_years	designation	classs	tier
72704	other	29a71dd13adf6d2d497571a565bb3096cf66cb46cd1ece...	1000150000	other	9.000065	7	1	1	1
117444	obvqnuqxdwgb	5b4bed51797140db4ed52018a979db1e34cee49e27b488...	255555555	other	8.407485	4	1	1	1
3294	other	06d231f167701592a69cdd7d5c825a0f5b30f0347a4078...	250000000	other	8.397940	1	1	1	1
16558	other	214035fc90945d84b9772dab3fce7fe328b96677d84bf0...	200000000	other	8.301030	3	1	1	1
16670	ogwxtnt stztqvrt srgmvr ogrhnxgz wtzntq	214bc79c4f76ac30da01be091f245dad007a47df640a2f...	200000000	other	8.301030	8	1	1	1
21605	fxuqg rxbxnta	89f343bf01094accb8b0b2c799499daf6bf881321db2e4...	200000000	data analyst	8.301030	5	1	1	1
7467	other	89e4f8e921ea205e2b5512cce828f634aa8214ef12f799...	200000000	other	8.301030	4	1	1	1
58208	stzuvwn	1c6bc8b95225bf255f939a64f9d60f84371f16eb621f3a...	200000000	other	8.301030	8	1	1	1
1082	vwwtznhqt	a071c4cd6d423e8d1841ba6133e6c4684f4eaba7dc1526...	200000000	backend engineer	8.301030	5	1	1	1
7486	other	52092435ab0f2a209a6f620d59a191c38160535ac5f8d0...	200000000	data analyst	8.301030	7	1	1	1

Q2. Top 10 employees of data science in Amazon / TCS etc earning more than their peers - Class 1

In [59]:
```
q2 = data5.loc[(data5['classs'] == 1) & (data5['job_position'] == 'data scientist')].sort_values(
    by=['ctc_log'], ascending=False).head(10)
q2
```

Out[59]:

	company_hash	email_hash	ctc	job_position	ctc_log	exp_years	designation	classs	tier
52724	zgzt	268a5aa92f0b6d0c675fc9cc1e300eb0c5930a3a139a23...	200000000	data scientist	8.301030	1	1	1	1
31235	ihvaqvnxw xzoxsyno ucn rna	bd222ea783ee372da4e0ad60fdccec0b8f37999a032025...	200000000	data scientist	8.301030	7	1	1	1
835	mqxonrtwgzt v bvyxzaqv sqghu wgbuvzj	cda8d723438e81185d2ee8c348870a4612eea974cdb2db...	200000000	data scientist	8.301030	5	1	1	1
2683	other	72ed7ced98573f71c8f95bc8b75aac4f0677e8872c6bec...	199800000	data scientist	8.300595	3	1	1	1
45132	pgnvp	ace1152ca60b6f2c62bb7c4a00bca0afd5a9bb2c297267...	150000000	data scientist	8.176091	21	1	1	1
36723	wrghaotp	a1223067ab5c4ff7fcf39ed4c053057f06090a57fc05ba...	127600000	data scientist	8.105851	5	1	1	1
1735	other	ee8dd42d6ea8365909147d861c7978d19f727a8075ba96...	102500000	data scientist	8.010724	2	1	1	1
10294	other	2e1d492bc09bfe0d4cc9757a9c63a296c1527af1c8ecc8...	100000000	data scientist	8.000000	1	1	1	1
151247	ntwy bvyxzaqv	6ad86d120e39db485331f9a0b2b1f15ce2a7bdaee778ab...	100000000	data scientist	8.000000	1	1	1	1
57251	other	259f6168edaed6bfb1d24bebe37fe7ddc8e6419884426e...	100000000	data scientist	8.000000	17	1	1	1

Q3. Bottom 10 employees of data science in Amazon / TCS etc earning less than their peers - Class 3

In [60]:
```
q3 = data5.loc[(data5['classs'] == 3) & (data5['job_position'] == 'data scientist')].sort_values(
    by=['ctc_log'], ascending=True).head(10)
q3
```

Out[60]:

	company_hash	email_hash	ctc	job_position	ctc_log	exp_years	designation	classs	tier
168056	other	05801a432a038c254972e356598ca6aa139a18c31d6611...	4000	data scientist	3.602060	1	3	3	3
8684	bxyhu wgbbhzxwvnxgz	690f6fdab1ab7514a6a9325ebd6cfe910dbf12d46b6fde...	4000	data scientist	3.602060	4	1	3	3
193877	other	585f7e9865dcdcaad7edf10909d796ba2c5210cde3530b...	4000	data scientist	3.602060	5	3	3	3
10810	srgmvrtast xzntrrxstzwt ge nyxzso	8001bc017fbe95541d23f5780c3edb988b7d9b2225e39e...	4000	data scientist	3.602060	5	1	3	3
50940	onhatzn	bd9c04a574090e05b366a81cdb2f3f565d0c60fa8b1647...	6000	data scientist	3.778151	1	3	3	3
136740	other	e374eea75640881206a21894f69190138c2c0535277dc1...	7000	data scientist	3.845098	5	3	3	3
24059	other	ab2dc9db23c3104f0b6b3dbd4cdd5bfb9e5829b8b7943d...	7200	data scientist	3.857332	5	3	3	3
182937	other	287dd26e9357888e0ba2c7482764131f7bbcb1748a4f56...	7250	data scientist	3.860338	3	3	3	3
92494	other	0dcbef1fe34438edbb39b52451378ea61ac2b84a56d919...	7500	data scientist	3.875061	5	3	3	3
9382	nvnv wgzohrnvzwj otqcxwto	3175d03fd4618eb293d6f5a1d13d42a0c79f68e9acaaa3...	7500	data scientist	3.875061	2	3	3	3

Q4. Bottom 10 employees (earning less than most of the employees in the company)- Tier 3

In [61]:
```
q4 = data5.loc[(data5['tier'] == 3)].sort_values(by=['ctc_log'], ascending=True).head(10)
q4
```

Out[61]:

	company_hash	email_hash	ctc	job_position	ctc_log	exp_years	designation	classs	tier
135212	xzntqcxtfmxn	3505b02549ebe2c95840ac6f0a35561a3b4cbe4b79cdb1...	2	backend engineer	0.301030	8	3	3	3
118044	xzntqcxtfmxn	f2b58aeed3c074652de2cfd3c0717a5d21d6fbcf342a78...	6	other	0.778151	9	3	3	3
113979	xzntqcxtfmxn	23ad96d6b6f1ecf554a52f6e9b61677c7d73d8a409a143...	14	other	1.146128	9	1	3	3
184648	other	b8a0bb340583936b5a7923947e9aec21add5ebc50cd60b...	15	other	1.176091	6	3	3	3
183506	other	75357254a31f133e2d3870057922feddeba82b88056a07...	16	other	1.204120	3	3	3	3
54722	other	8786759b95d673466e94f62f1b15e4f8c6bd7de6164074...	24	other	1.380211	2	3	3	3
91393	other	512f761579fb116e215cabc9821c7f81153f0763e16018...	25	android engineer	1.397940	6	3	3	3
116756	other	f7e5e788676100d7c4146740ada9e2f8974defc01f571d...	200	other	2.301030	1	3	3	3
166129	other	c411a6917058b50f44d7c62751be9b232155b23211de4c...	300	database administrator	2.477121	9	3	3	3
81891	other	edcfb902656b736e1f35863298706d9d34ee795b7ed85a...	500	cofounder	2.698970	4	3	3	3

Q5. Top 10 companies (based on their CTC)

In [62]:
```
q5 = data5.groupby(by = 'company_hash')['ctc'].mean().round(2).reset_index().sort_values(by = 'ctc',ascending = False).head(10)
q5
```

Out[62]:

	company_hash	ctc
2798	xzaxvmhrro	53741428.57
1197	obvqnuqxdwgb	43952592.50
666	ho tzsxzttqxzs wgbuvzj	43487142.86
532	fgqraihvzn rrw	34958333.33
423	egdwgzz	34232500.00
1273	omx	34123333.33
2748	xqgz bghznvxz	33958333.33
1070	ntvwy egq xzaxv	32327714.29
1177	nyt sqtvn wghqoto	32152000.00
1528	psxor	31945600.00

Q6. Top 2 positions in every company (based on their CTC)

In [63]:
```
q6 = pd.pivot_table(data=data5, values='ctc', index=['company_hash','job_position'],
    aggfunc='mean', fill_value=None, margins=False, dropna=True, sort=True).round(2).reset_index()
q6 = q6.groupby(by=['company_hash']).apply(lambda x:x.sort_values(by='ctc', ascending = False).head(2).reset_index())
q6 = q6.reset_index(drop = True)
q6['ctc'] = q6['ctc'].astype('int')
q6.head(10)
```

Out[63]:

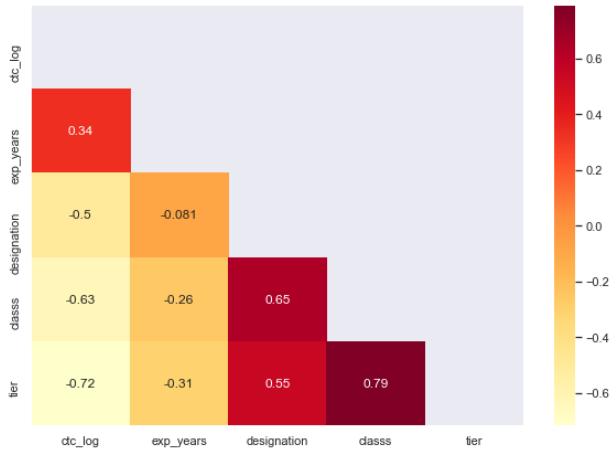
	index	company_hash	job_position	ctc
0	4	a ntwyzgrgsxto	other	1815625
1	1	a ntwyzgrgsxto	backend engineer	850000
2	8	aaqxctz avnv owxtzwto vzvrjnxwo ucn rna	other	2050000
3	5	aaqxctz avnv owxtzwto vzvrjnxwo ucn rna	backend engineer	730000
4	11	adw ntwyzgrgsj	data analyst	20420000
5	21	adw ntwyzgrgsj	product manager	5000000
6	39	adw ntwyzgrgsxto	product designer	100000000
7	36	adw ntwyzgrgsxto	fullstack engineer	8274538
8	44	agdutq	backend architect	2500000
9	45	agdutq	backend engineer	1525000

Unsupervised learning - Clustering

In [64]:
```
data5.ctc = data5.ctc.clip(data.ctc.quantile(0.01), data.ctc.quantile(0.99))
```

In [65]:
```
X = data5[['ctc_log', 'exp_years', 'designation', 'classs', 'tier']].copy()
```

In [66]:
```
##Correlation Matrix
corr = X.corr(method='spearman')
mask = np.triu(corr)
plt.figure(figsize=(10,7))
sns.heatmap(corr, annot = True, mask = mask, cmap = 'YlOrRd')
plt.show()
```



In [67]:
```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
```

```
X_std = sc.fit_transform(X)
X_std= pd.DataFrame(X_std, columns=X.columns, index=X.index)
X_std.head()
```

Out[67]:

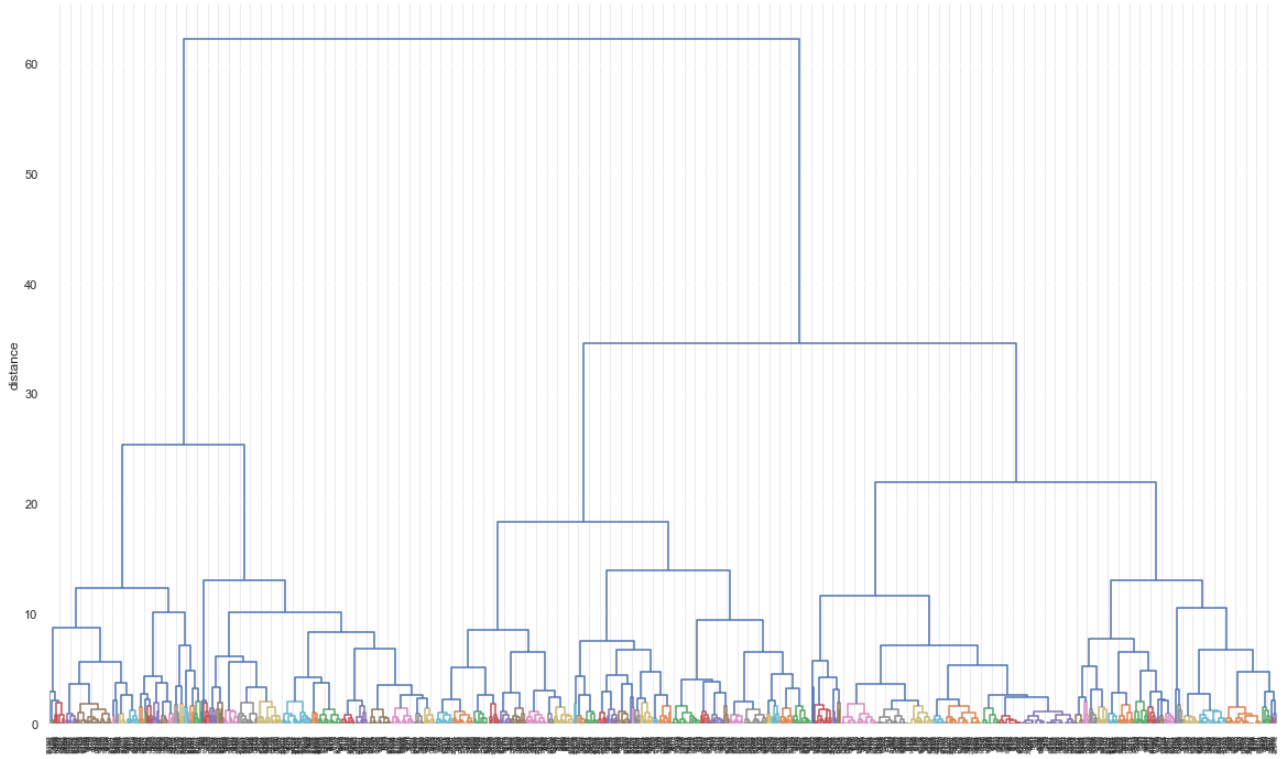
	ctc_log	exp_years	designation	classs	tier
0	0.156150	-0.210545	-1.016896	-1.302839	-0.252074
1	-0.690505	-0.702764	1.211433	1.005209	0.938476
2	0.722441	0.035565	-1.016896	-1.302839	-1.442625
3	-0.271985	-0.456654	1.211433	1.005209	0.938476
4	0.384587	-0.456654	-1.016896	-1.302839	-1.442625

In [68]:
```
X_std.shape
```

Out[68]:
```
(205517, 5)
```

In [69]:
```
import scipy.cluster.hierarchy as sch
import matplotlib.pyplot as plt

sample = X_std.sample(1000)
Z = sch.linkage(sample, method='ward')

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

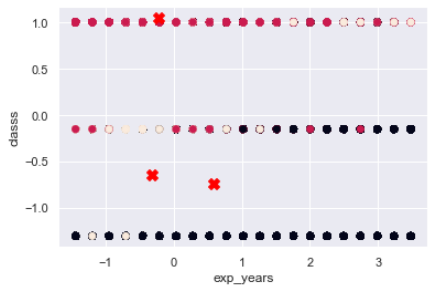
Out[69]:
```
Text(0, 0.5, 'distance')
```



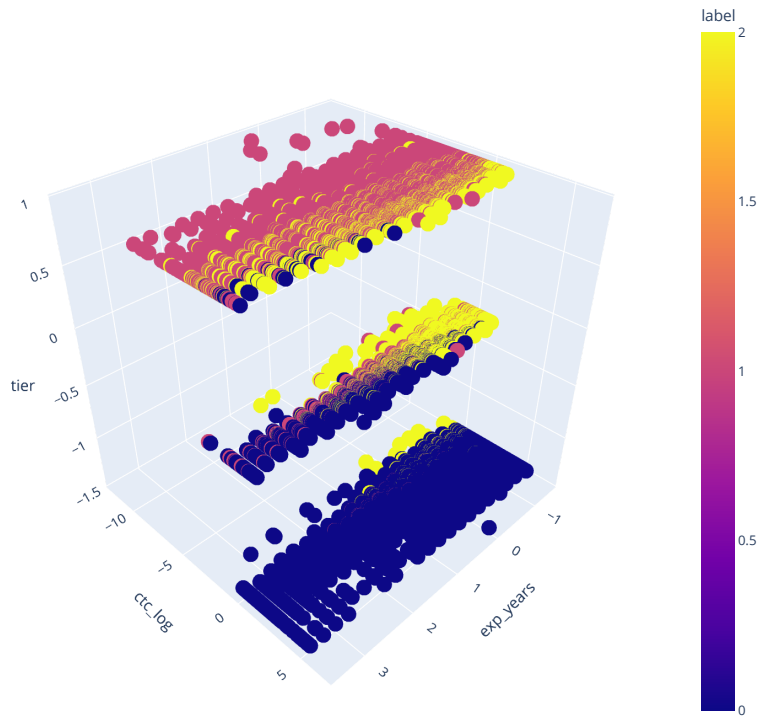
In [70]:
```
from sklearn.cluster import KMeans

k = 3
kmeans = KMeans(n_clusters=k, random_state=42)
y_pred = kmeans.fit_predict(X_std)

##coordinates of the cluster centers
# kmeans.cluster_centers_
clusters = pd.DataFrame(X_std, columns=X.columns)
clusters['label'] = kmeans.labels_
```

In [71]:
```
x_axis = 'exp_years'
y_axis = 'classs'

plt.scatter(clusters[x_axis], clusters[y_axis], c=clusters['label'])
plt.scatter(kmeans.cluster_centers_[:, 1], kmeans.cluster_centers_[:, 2], color="red", marker="X", s=100)
plt.xlabel(x_axis)
plt.ylabel(y_axis)
```

Out[71]:
```
Text(0, 0.5, 'classs')
```



In [72]:
```
import plotly.express as px
fig = px.scatter_3d(clusters, x='exp_years', y='ctc_log', z='tier', color='label', width=800, height=800)
fig.show()
```



Insights, Methodology & Recommendations

- Initial Data had ~ 1.88L rows and 5 relevant columns including company_hash, orgyear, ctc, job_position, ctc_updated_year
- Unique Job Positions after preprocessing: 856. "backend engineer" has highest frequency : 40298, followed by "fullstack engineer" : 24030 & "frontend engineer": 10102. This indicated majorly IT-domain based responses are covered in this dataset.
- Unique company_hash after preprocessing: 37180. Company "nvnv wgzohrnvzwj otqcxwto" has highest frequency : 4282. Followed by "xzegojo" : 3043 & "vbvkgz" : 3004.
- Numerical columns - orgyear & ctc were clipped at 1 percentile & 99 percentile, to remove outliers.
- Since "ctc" was highly right skewed distribution, log10 was taken, "ctc_log" was created and considered for all further clustering.
- Column "exp_years" was created to map experience of employee. Range of experience was clipped from 1 to 22 years.
- At the start, no clear clusters were observed between "exp_years" & "ctc_log". Hence other columns were added to created clusters.
- Manual Clustering was created by grouping on three levels.
    - designation: Company, job-profile & years of experience
    - classs : Company, job-profile
    - tier : Company
- Using Hierarchial Clustering, 3 optimal number of clusters were identified. Dendogram was created accordingly.
- Using k = 3 & KMeans Clustering algorithm, data was clusterred in 3 clusters.
- Clusters are not very well separable.
    - However, tier has strong correlation with CTC. People lying in Tier-3 have low ctc irrespective of years of experience.
    - This indicates, they are part of low paying companies.
- It is recommended not to take major business decisions based on the conclusions provided.

---------- END----------

In [ ]: