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 re
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (12,8)

import seaborn as sns
```

In [2]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
df = pd.read_csv('/Users/bose/Desktop/scaler_clustering.csv', index_col=0)
```

In [4]:
```
df.sample(10)
```

Out[4]:

	company_hash	email_hash	orgyear	
198726	hmtq	f2f0296a9a833add9bd3821a8138a68ea0b4650e4f8d0e...	2015.0	43
204604	eqtoytq	bc7d30870f28439dab43bb2344c298b6959bf1c9cfce4f...	2019.0	21
200507	nvnv wgzohrnvzwj otqcxwto	83adc9a6e84bbfa62c9ef368228fd42faae98691cb5f55...	2017.0	3
68895	ywr ntwyzgrgsxto	07e91866ce776ff766c8bc2bd924f9e01b0832720443b8...	2020.0	4
41564	vbagwo	655df8b33a384ff6c626f4ada054e411e87d0af20a3bd1...	2019.0	8
15373	ovo	f020afbbece4a6b463d591cf92038ffd3b9e94dabc7502...	2009.0	15
185869	ofxssj	704b1ecb19ed94de06da99eef285abbf35bd2bfa57d2fb...	2015.0	29
118848	xcj wgbuntwy	3f8243066a30b25a2c446364b7fe04d09ede95224b5857...	2019.0	9
171765	ctqxkgz	f1d3f76adf959e94e75114e1ffdbcc3461b8ad00c3f9fa...	2012.0	9
139089	gqvwrt	cbb9a751042f611e7f22208edbd09e5596f7b2143e4b7b...	2017.0	7

In [5]:
```
df.shape
```

Out[5]:
```
(205843, 6)
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 205843 entries, 0 to 206922
Data columns (total 6 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   company_hash      205799 non-null  object 
 1   email_hash        205843 non-null  object 
 2   orgyear           205757 non-null  float64
 3   ctc               205843 non-null  int64  
 4   job_position      153281 non-null  object 
 5   ctc_updated_year  205843 non-null  float64
dtypes: float64(2), int64(1), object(3)
memory usage: 11.0+ MB
```

In [7]:
```
df.isna().sum()
```

Out[7]:
```
company_hash           44
email_hash              0
orgyear                86
ctc                     0
job_position        52562
ctc_updated_year        0
dtype: int64
```

In [8]:
```
df.isna().sum()/len(df)*100
```

Out[8]:
```
company_hash         0.021376
email_hash           0.000000
orgyear              0.041779
ctc                  0.000000
job_position        25.534995
ctc_updated_year     0.000000
dtype: float64
```

In [9]:
```
df.describe()
```

Out[9]:

	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
50%	2016.000000	9.500000e+05	2020.000000
75%	2018.000000	1.700000e+06	2021.000000
max	20165.000000	1.000150e+09	2021.000000

In [10]:
```
df.describe(include='object')
```

Out[10]:

	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

In [11]:
```
def preprocess_string(string):
 new_string= re.sub('[^A-Za-z ]+', '', string).lower().strip()
 return new_string

mystring='\tAirtel\\\\&&**() X Labs'
preprocess_string(mystring)
```

Out[11]:
```
'airtel x labs'
```

In [12]:
```
df["company_hash"].nunique()
```

Out[12]:
```
37299
```

In [13]:
```
df["company_hash"] = df["company_hash"].apply(lambda x: preprocess_string(st
df["company_hash"].nunique()
```

Out[13]:
```
37208
```

In [14]:
```
df["job_position"].nunique()
# 1017 unique job positions are there in the dataset
```

Out[14]:
```
1017
```

In [15]:
```
df["job_position"] = df["job_position"].apply(lambda x: preprocess_string(st
df["job_position"].nunique()
```

Out[15]:
```
857
```

In [16]:
```
df.drop("email_hash",axis = 1,inplace=True)
```

In [17]:
```
df.duplicated().sum()
```

Out[17]:
```
17597
```

In [18]:
```
df.isna().sum()
```

Out[18]:
```
company_hash          0
orgyear              86
ctc                   0
job_position          0
ctc_updated_year      0
dtype: int64
```

In [19]:
```
(df["company_hash"] == "").sum()
```

Out[19]:
```
89
```

In [20]:
```
(df["company_hash"] == "nan").sum()
```

Out[20]:
```
44
```

In [21]:
```
(df["job_position"] == "").sum()
```

Out[21]:
```
9
```

In [22]:
```
(df["job_position"] == "nan").sum()
```

Out[22]:
```
52562
```

In [23]:
```
# removing the records where company_hash or job_position reocords are not a
```

In [24]:
```
df[(df["company_hash"] == "") | (df["job_position"] == "")].sample(10)
```

Out[24]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year
197495		2020.0	2700000	nan	2019.0
47284		2019.0	370000	engineering leadership	2021.0
188918		2019.0	900000	nan	2020.0
43486		2021.0	800000	frontend engineer	2021.0
156894		2020.0	700000	nan	2020.0
167717		2018.0	1500000	backend engineer	2020.0
200817		2015.0	200000	fullstack engineer	2019.0
197978		2020.0	1000000	nan	2019.0
76907		2021.0	800000	nan	2021.0
129318		2018.0	11000	backend engineer	2019.0

In [25]:
```
len(df[(df["company_hash"] == "") | (df["job_position"] == "")])
```

Out[25]:
```
98
```

In [26]:
```
df = df[~((df["company_hash"] == "") | (df["job_position"] == ""))]
df
```

Out[26]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0
2	ojzwnvwnxw vx	2015.0	2000000	backend engineer	2020.0
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0
...	...	...	...	...	...
206918	vuurt xzw	2008.0	220000	nan	2019.0
206919	husqvawgb	2017.0	500000	nan	2020.0
206920	vwwgrxnt	2021.0	700000	nan	2021.0
206921	zgn vuurxwvmrt	2019.0	5100000	nan	2019.0
206922	bgqsvz onvzrtj	2014.0	1240000	nan	2016.0

205745 rows × 5 columns

Data Preprocessing

In [27]:
```
df["orgyear"].isna().sum()
```

Out[27]:
```
86
```

Imputing Employee Start Year as per the median year as per each company.

In [28]:
```
df.groupby("company_hash")["orgyear"].transform("median")
```

Out[28]:
```
0         2014.0
1         2016.0
2         2015.0
3         2016.0
4         2017.0
           ...
206918    2018.0
206919    2017.0
206920    2016.0
206921    2020.0
206922    2015.0
Name: orgyear, Length: 205745, dtype: float64
```

In [29]:
```
df["orgyear"].fillna(df['orgyear'].isnull().sum(),inplace=True)
```

In [30]:
```
df["orgyear"].isna().sum()
```

Out[30]:
```
0
```

In [31]:
```
df.head()
```

Out[31]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0
2	ojzwnvwnxw vx	2015.0	2000000	backend engineer	2020.0
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0

Outliers Treatment

orgyear

In [32]:
```
df["orgyear"].value_counts()
```

Out[32]:
```
2018.0    25240
2019.0    23402
2017.0    23237
2016.0    23038
2015.0    20602
          ...
2107.0        1
1972.0        1
2101.0        1
208.0         1
200.0         1
Name: orgyear, Length: 78, dtype: int64
```

In [33]:
```
df["orgyear"].quantile(0.001)
```

Out[33]:
```
1990.0
```

In [34]:
```
df["orgyear"].quantile(0.999)
```

Out[34]:
```
2023.0
```

In [35]:
```
df["orgyear"] = df["orgyear"].clip(1990,2022)
```

ctc updated year

In [36]:
```
df["ctc_updated_year"].quantile(0.001)
```

Out[36]:
```
2015.0
```

In [37]:
```
df["ctc_updated_year"].quantile(0.99)
```

Out[37]:
```
2021.0
```

ctc

In [38]:
```
df["ctc"].quantile(0.01)
```

Out[38]:
```
37000.0
```

In [39]:
```
df["ctc"].quantile(0.999)
```

Out[39]:
```
200000000.0
```

In [40]:
```
df = df.loc[((df.ctc) > df.ctc.quantile(0.01)) & ((df.ctc) < df.ctc.quantile
```

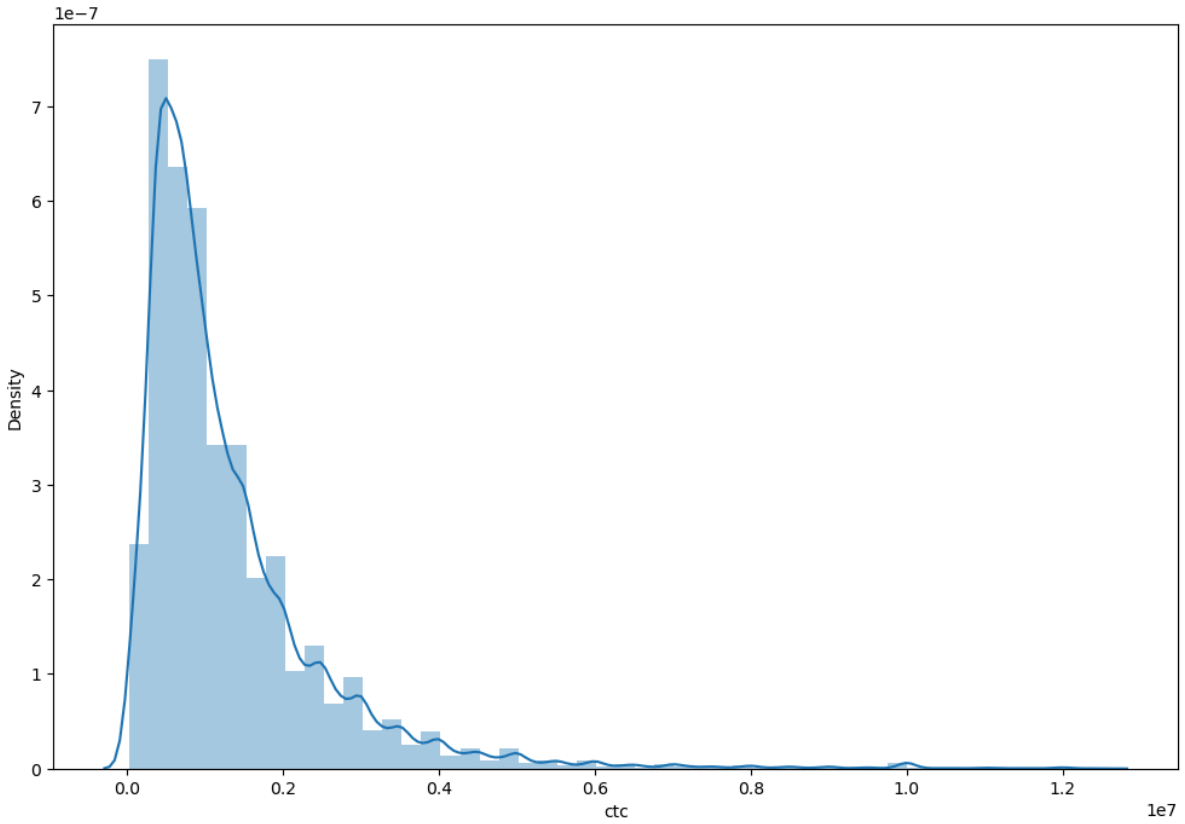
In [41]:
```
df.head()
```

Out[41]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0
2	ojzwnvwnxw vx	2015.0	2000000	backend engineer	2020.0
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0

In [42]:
```
sns.distplot(df["ctc"])
```

Out[42]:
```
<AxesSubplot:xlabel='ctc', ylabel='Density'>
```



In [43]:
```
df.loc[df['job_position']=='nan', 'job_position']=np.nan
```

In [44]:
```
df.loc[df["company_hash"]=="nan","company_hash"] = np.nan
```

Feature Engineering

In [45]:
```
df.loc[df.groupby("company_hash")["ctc"].transform("count") < 5,"company_has
```

In [46]:
```
(df["company_hash"] == "Others").sum()
```

Out[46]:
```
46434
```

In [47]:
```
df['orgyear'].describe()
```

Out[47]:
```
count    201625.000000
mean       2015.104769
std           4.256063
min        1990.000000
25%        2013.000000
50%        2016.000000
75%        2018.000000
max        2022.000000
Name: orgyear, dtype: float64
```

In [48]:
```
# making years of experience column
df["years_of_experience_in_organization"] = 2023 - df["orgyear"]
```

In [49]:
```
df.head()
```

Out[49]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience_i
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	
2	Others	2015.0	2000000	backend engineer	2020.0	
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	

In [50]:
```
df.duplicated().sum()
```

Out[50]:
```
37683
```

In [51]:
```
df.drop_duplicates(inplace=True)
df.shape
```

Out[51]:
```
(163942, 6)
```

In [52]:
```
df.isna().sum()
```

Out[52]:
```
company_hash                          42
orgyear                                0
ctc                                    0
job_position                       36745
ctc_updated_year                       0
years_of_experience_in_organization    0
dtype: int64
```

In [53]:
```
# records having ctc_updated_year higher than their organization joining yea
(df["ctc_updated_year"] < df["orgyear"]).sum()
```

Out[53]:
```
7181
```

In [54]:
```
df.ctc_updated_year = df[["ctc_updated_year","orgyear"]].max(axis = 1)
```

In [55]:
```
(df["ctc_updated_year"] < df["orgyear"]).sum()
```

Out[55]:
```
0
```

In [56]:
```
df.head()
```

Out[56]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience_i
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	
2	Others	2015.0	2000000	backend engineer	2020.0	
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	

Filling null values with others

In [57]:
```
df['job_position'] = df['job_position'].fillna('Others')
df['company_hash'] = df['company_hash'].fillna('Others')
```

In [58]:
```
df.isna().sum()
```

Out[58]:
```
company_hash                          0
orgyear                               0
ctc                                   0
job_position                          0
ctc_updated_year                      0
years_of_experience_in_organization    0
dtype: int64
```

In [59]:
```
df.duplicated().sum()
```

Out[59]:
```
1061
```

In [60]:
```
df.describe()
```

Out[60]:

	orgyear	ctc	ctc_updated_year	years_of_experience_in_organizatio
count	163942.000000	1.639420e+05	163942.000000	163942.00000
mean	2014.772218	1.425498e+06	2019.595540	8.22778
std	4.402053	1.303985e+06	1.334962	4.40205
min	1990.000000	3.800000e+04	2015.000000	1.00000
25%	2013.000000	6.000000e+05	2019.000000	5.00000
50%	2016.000000	1.039999e+06	2020.000000	7.00000
75%	2018.000000	1.800000e+06	2021.000000	10.00000
max	2022.000000	1.250000e+07	2022.000000	33.00000

In [61]:
```
df.columns
```

Out[61]:
```
Index(['company_hash', 'orgyear', 'ctc', 'job_position', 'ctc_updated_yea
r',
       'years_of_experience_in_organization'],
      dtype='object')
```

Manual Clustering based on Company, Job Position and Years of Experience

In [62]:
```
GROUPED_CTC = df.groupby(["years_of_experience_in_organization", "job_posit
```

In [63]:
```
GROUPED_CTC
```

Out[63]:

			count	mean	
years_of_experience_in_organization	job_position	company_hash			
1.0	Others	Others	58.0	1.586207e+06	2.0
		agzn fgqp xz vzj gqsvzxkvnxgz	1.0	1.600000e+06	
		atrgxnnt	1.0	1.000000e+06	
		atrr	1.0	1.000000e+06	
		atrr ntwyzgrgsxto	2.0	1.000000e+06	2.8
...	...	...	...	...	
33.0	qa engineer	hzxntaytvrny sqghu	1.0	5.400000e+05	
		tmxd ogenfvqt xzaxv ucn rna	1.0	1.220000e+06	
		utrvnqg ogrhnxgzo ucnrna	1.0	6.000000e+05	
	research engineers	ovbohzs qa xzonxnhnt xzaxv atryx	1.0	1.400000e+06	
	support engineer	Others	2.0	3.700000e+05	3.2

66191 rows × 8 columns

In [64]:
```
df_GROUPED_CTC_BY_E_P_C = df.merge(GROUPED_CTC,
 on = ["years_of_experience_in_organization",
 "job_position",
 "company_hash"],
 how = "left")
```

In [65]:
```
df_GROUPED_CTC_BY_E_P_C
```

Out[65]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experi
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	
2	Others	2015.0	2000000	backend engineer	2020.0	
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	
...	...	...	...	...	...	
163937	vuurt xzw	2008.0	220000	Others	2019.0	
163938	husqvawgb	2017.0	500000	Others	2020.0	
163939	vwwgrxnt	2021.0	700000	Others	2021.0	
163940	zgn vuurxwvmrt	2019.0	5100000	Others	2019.0	
163941	bgqsvz onvzrtj	2014.0	1240000	Others	2016.0	

163942 rows × 14 columns

In [67]:
```
def classification(x,ctc_50,ctc_75):
 if x < ctc_50:
     return 3
 elif x >= ctc_50 and x <= ctc_75:
     return 2
 elif x >= ctc_75:
     return 1
```

whichever learner has ctc compared to their years of experience , respective company , position

giving designation as 3 when ctc is < 50th percentile in his position ,experience and company

giving designation as 2 when ctc is between 50th and 75th percentile in his position ,experience and company

giving designation as 1 when ctc is > 75th percentile in his position ,experience and company

In [68]:
```
df_GROUPED_CTC_BY_E_P_C["designation_in_organization"] = df_GROUPED_CTC_BY_E
```

In [69]:
```
df_GROUPED_CTC_BY_E_P_C
```

Out[69]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experi
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	
2	Others	2015.0	2000000	backend engineer	2020.0	
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	
...	...	...	...	...	...	
163937	vuurt xzw	2008.0	220000	Others	2019.0	
163938	husqvawgb	2017.0	500000	Others	2020.0	
163939	vwwgrxnt	2021.0	700000	Others	2021.0	
163940	zgn vuurxwvmrt	2019.0	5100000	Others	2019.0	
163941	bgqsvz onvzrtj	2014.0	1240000	Others	2016.0	

163942 rows × 15 columns

In [70]:
```
df_GROUPED_CTC_BY_E_P_C.designation_in_organization.value_counts(normalize=T
```

Out[70]:
```
2    0.456393
3    0.331660
1    0.211947
Name: designation_in_organization, dtype: float64
```

In [71]:
```
df_GROUPED_CTC_BY_E_P_C
```

Out[71]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experi
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	
2	Others	2015.0	2000000	backend engineer	2020.0	
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	
...	...	...	...	...	...	
163937	vuurt xzw	2008.0	220000	Others	2019.0	
163938	husqvawgb	2017.0	500000	Others	2020.0	
163939	vwwgrxnt	2021.0	700000	Others	2021.0	
163940	zgn vuurxwvmrt	2019.0	5100000	Others	2019.0	
163941	bgqsvz onvzrtj	2014.0	1240000	Others	2016.0	

163942 rows × 15 columns

In [72]:
```
df_GROUPED_CTC_BY_E_P_C.drop(columns=['count',
 'mean',
 'std',
'min',
'25%',
'50%',
'75%',
'max'],axis = 1,inplace=True)
```

In [73]:
```
df_GROUPED_CTC_BY_E_P_C
```

Out[73]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experi
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	
2	Others	2015.0	2000000	backend engineer	2020.0	
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	
...	...	...	...	...	...	
163937	vuurt xzw	2008.0	220000	Others	2019.0	
163938	husqvawgb	2017.0	500000	Others	2020.0	
163939	vwwgrxnt	2021.0	700000	Others	2021.0	
163940	zgn vuurxwvmrt	2019.0	5100000	Others	2019.0	
163941	bgqsvz onvzrtj	2014.0	1240000	Others	2016.0	

163942 rows × 7 columns

In [74]:
```
df_GROUPED_CTC_BY_E_P_C.shape
```

Out[74]:
```
(163942, 7)
```

Manual Clustering on Company and Job Position

In [75]:
```
GROUPED_C_J=df.groupby(['job_position','company_hash'])['ctc'].describe()
GROUPED_C_J
```

Out[75]:

job_position	company_hash	count	mean	std	min	25%	
Others	Others	3520.0	1.366188e+06	1.445330e+06	40000.0	409999.0	900
	a ntwyzgrgsxto	6.0	1.229167e+06	1.401465e+06	350000.0	518750.0	587
	aaqxctz avnv owxtzwto vzvrjnxwo ucn rna	1.0	5.000000e+05	NaN	500000.0	500000.0	500
	abwavnv ojontb	1.0	7.000000e+05	NaN	700000.0	700000.0	700
	adw ntwyzgrgsj	69.0	8.502319e+05	1.036041e+06	80000.0	380000.0	500
...	...	...	...	...	...	...	
wordpress developer	Others	1.0	6.000000e+05	NaN	600000.0	600000.0	600
worker	zgn vuurxwvmrt vwwghzn	1.0	2.000000e+05	NaN	200000.0	200000.0	200
x	Others	1.0	4.000000e+05	NaN	400000.0	400000.0	400
young professional ii	sgctqzbtzn ge xzaxv	1.0	5.000000e+05	NaN	500000.0	500000.0	500
zomato	kgbvng	2.0	3.000000e+05	2.828427e+05	100000.0	200000.0	300

25593 rows × 8 columns

In [76]:
```
df_GROUPED_C_J=df.merge(GROUPED_C_J, on=['job_position','company_hash'], how
```

In [77]:
```
df_GROUPED_C_J.head()
```

Out[77]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience_i
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	
2	Others	2015.0	2000000	backend engineer	2020.0	
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	

In [78]:
```
df_GROUPED_C_J['classs'] = df_GROUPED_C_J.apply(lambda x: classification(x[
```

In [79]:
```
df_GROUPED_C_J.head()
```

Out[79]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience_i
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	
2	Others	2015.0	2000000	backend engineer	2020.0	
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	

In [80]:
```
df_GROUPED_C_J.classs.value_counts(normalize=True)
```

Out[80]:
```
3    0.435373
2    0.320101
1    0.244526
Name: classs, dtype: float64
```

In [81]:
```
df_GROUPED_C_J.drop(columns=['count',
 'mean',
 'std',
'min',
'25%',
'50%',
'75%',
'max'],axis = 1,inplace=True)
```

In [82]:
```
df_GROUPED_CTC_BY_E_P_C.iloc[0]
```

Out[82]:
```
company_hash                              atrgxnnt xzaxv
orgyear                                          2016.0
ctc                                             1100000
job_position                                      other
ctc_updated_year                                 2020.0
years_of_experience_in_organization                 7.0
designation_in_organization                           2
Name: 0, dtype: object
```

In [83]:
```
df_GROUPED_C_J.iloc[0]
```

Out[83]:
```
company_hash                              atrgxnnt xzaxv
orgyear                                          2016.0
ctc                                             1100000
job_position                                      other
ctc_updated_year                                 2020.0
years_of_experience_in_organization                 7.0
classs                                                1
Name: 0, dtype: object
```

In [84]:
```
df_Grouped = df_GROUPED_CTC_BY_E_P_C.merge(df_GROUPED_C_J, on=['company_hash
 'orgyear',
'ctc',
'job_position',
'years_of_experience_in_organization',
'ctc_updated_year'], how='left')
```

In [85]:
```
df_Grouped.head()
```

Out[85]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience_i
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	
2	Others	2015.0	2000000	backend engineer	2020.0	
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	

In [86]:
```
df_Grouped.shape
```

Out[86]:
```
(166228, 8)
```

Manual Clustering based on company

based on ctc per company , assigning company as tier 1 2 and 3 per each learners

In [88]:
```
GROUPED_C = df.groupby(['company_hash'])['ctc'].describe()
df_company = df.merge(GROUPED_C, on=['company_hash'], how='left')
```

In [89]:
```
df_company.head()
```

Out[89]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience_i
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	
2	Others	2015.0	2000000	backend engineer	2020.0	
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	

In [90]:
```
df_company['tier'] =df_company.apply(lambda x: classification(x['ctc'],x['50
```

In [91]:
```
df_company.tier.value_counts(normalize=True)
```

Out[91]:
```
3    0.477364
2    0.282911
1    0.239725
Name: tier, dtype: float64
```

In [92]:
```
df_company.drop(['count','mean','std','min','25%','50%','75%','max'],
 axis = 1,
 inplace=True)
```

In [93]:
```
df_company.iloc[0]
```

Out[93]:
```
company_hash                              atrgxnnt xzaxv
orgyear                                          2016.0
ctc                                             1100000
job_position                                      other
ctc_updated_year                                 2020.0
years_of_experience_in_organization                 7.0
tier                                                  2
Name: 0, dtype: object
```

In [94]:
```
df_Grouped.iloc[0]
```

Out[94]:
```
company_hash                              atrgxnnt xzaxv
orgyear                                          2016.0
ctc                                             1100000
job_position                                      other
ctc_updated_year                                 2020.0
years_of_experience_in_organization                 7.0
designation_in_organization                           2
classs                                                1
Name: 0, dtype: object
```

In [95]:
```
df_Grouped = df_Grouped.merge(df_company,
 on=['company_hash',
 'orgyear','ctc',
'job_position',
'years_of_experience_in_organization',
'ctc_updated_year'
 ])
```

In [96]:
```
df_Grouped
```

Out[96]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experie
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	
2	Others	2015.0	2000000	backend engineer	2020.0	
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	
...	...	...	...	...	...	
171311	vuurt xzw	2008.0	220000	Others	2019.0	
171312	husqvawgb	2017.0	500000	Others	2020.0	
171313	vwwgrxnt	2021.0	700000	Others	2021.0	
171314	zgn vuurxwvmrt	2019.0	5100000	Others	2019.0	
171315	bgqsvz onvzrtj	2014.0	1240000	Others	2016.0	

171316 rows × 9 columns

In [97]:
```
X = df_Grouped.copy()
```

In [98]:
```
X.shape
```

Out[98]: (171316, 9)

In [101…
```
X_data = X.drop(["company_hash","job_position"],axis = 1)
```

Final data for Model

In [102…
```
X_data
```

Out[102]:

	orgyear	ctc	ctc_updated_year	years_of_experience_in_organization	designa
0	2016.0	1100000	2020.0	7.0	
1	2018.0	449999	2019.0	5.0	
2	2015.0	2000000	2020.0	8.0	
3	2017.0	700000	2019.0	6.0	
4	2017.0	1400000	2019.0	6.0	
...	...	...	...	...	
171311	2008.0	220000	2019.0	15.0	
171312	2017.0	500000	2020.0	6.0	
171313	2021.0	700000	2021.0	2.0	
171314	2019.0	5100000	2019.0	4.0	
171315	2014.0	1240000	2016.0	9.0	

171316 rows × 7 columns

Standardization

In [103…
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_data)
X_sc = pd.DataFrame(scaler.transform(X_data), columns=X_data.columns, index=
```

In [104…
```
X_sc
```

Out[104]:

	orgyear	ctc	ctc_updated_year	years_of_experience_in_organization	desi
0	0.229439	-0.238430	0.298195	-0.229439	
1	0.680950	-0.741765	-0.452799	-0.680950	
2	0.003683	0.458493	0.298195	-0.003683	
3	0.455194	-0.548174	-0.452799	-0.455194	
4	0.455194	-0.006122	-0.452799	-0.455194	
...	...	...	...	...	
171311	-1.576605	-0.919866	-0.452799	1.576605	
171312	0.455194	-0.703046	0.298195	-0.455194	
171313	1.358216	-0.548174	1.049190	-1.358216	
171314	0.906705	2.859008	-0.452799	-0.906705	
171315	-0.222072	-0.130020	-2.705782	0.222072	

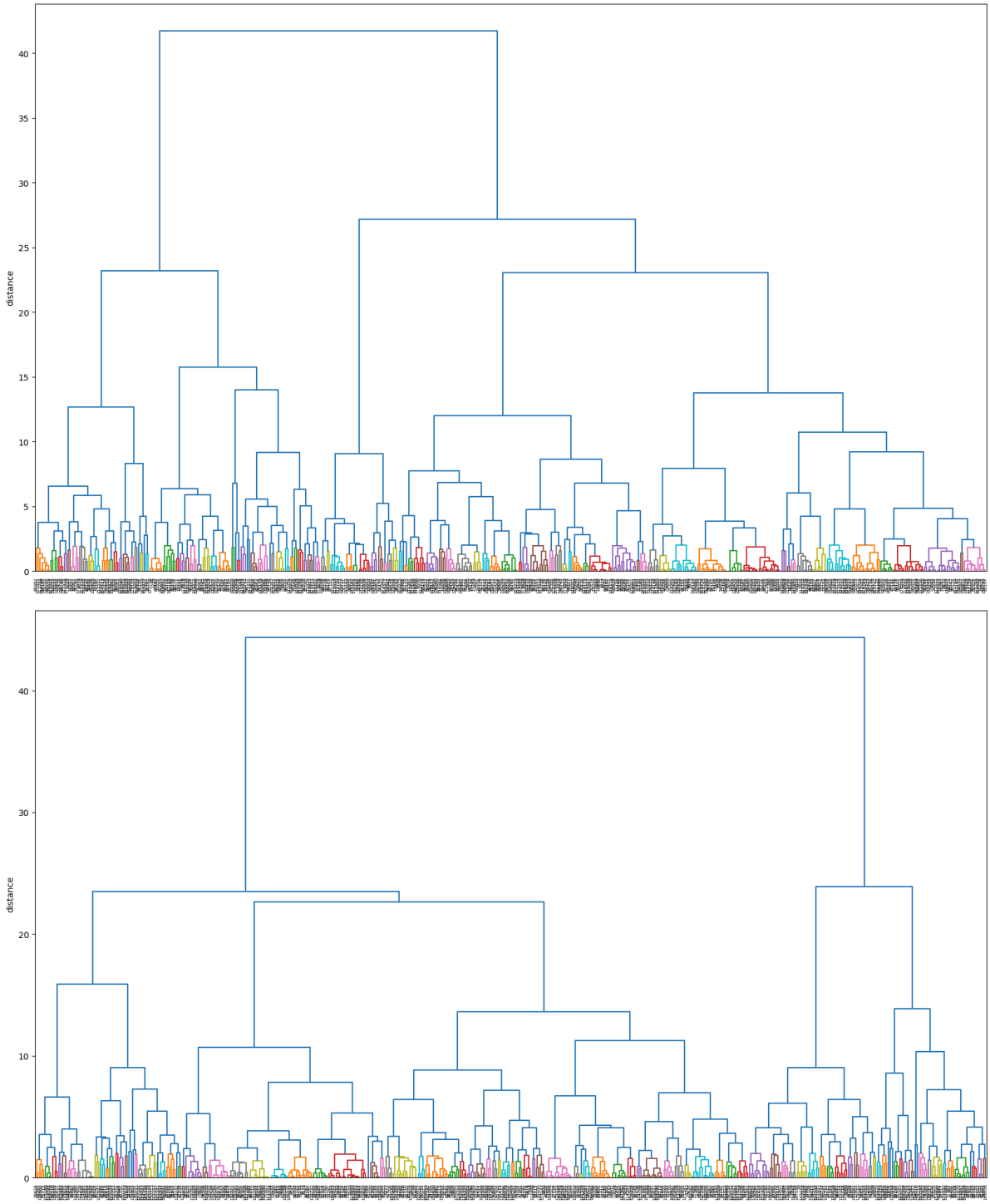
171316 rows × 7 columns

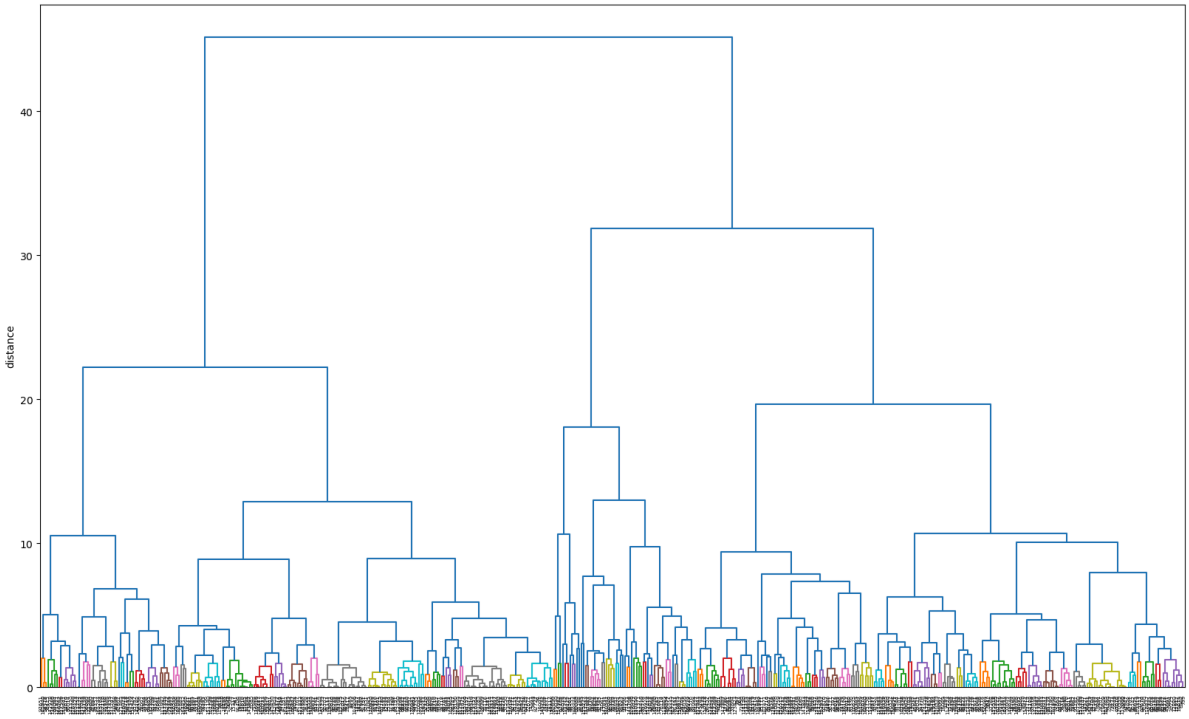
Hierarchial Clsutering

trying to get a high level idea about how many clusters we can from, by taking sample of 500 learners multiple times and forming hierarchy and visualising in dendrogram.

In [106…

```
import scipy.cluster.hierarchy as sch
import matplotlib.pyplot as plt
sample = X_sc.sample(500)
Z = sch.linkage(sample, method='ward')
fig, ax1 = plt.subplots(figsize=(20, 12))
sch.dendrogram(Z, labels=sample.index, ax=ax1, color_threshold=2)
plt.xticks(rotation=90)
ax1.set_ylabel('distance')
plt.show()
import scipy.cluster.hierarchy as sch
import matplotlib.pyplot as plt
sample = X_sc.sample(500)
Z = sch.linkage(sample, method='ward')
fig, ax2 = plt.subplots(figsize=(20, 12))
sch.dendrogram(Z, labels=sample.index, ax=ax2, color_threshold=2)
plt.xticks(rotation=90)
ax2.set_ylabel('distance')
plt.show()
import scipy.cluster.hierarchy as sch
import matplotlib.pyplot as plt
sample = X_sc.sample(500)
Z = sch.linkage(sample, method='ward')
fig, ax3 = plt.subplots(figsize=(20, 12))
sch.dendrogram(Z, labels=sample.index, ax=ax3, color_threshold=2)
plt.xticks(rotation=90)
ax3.set_ylabel('distance')
plt.show()
```


distance
0
5
10
15
20
25
30
35
40
distance
0
10
20
30
40



Based on dendrogram , we can observe there are 3 clusters in the data based on similarity

Further checking appropriate number of clusters using Elbow Method using k-Means clustering :

KMeans

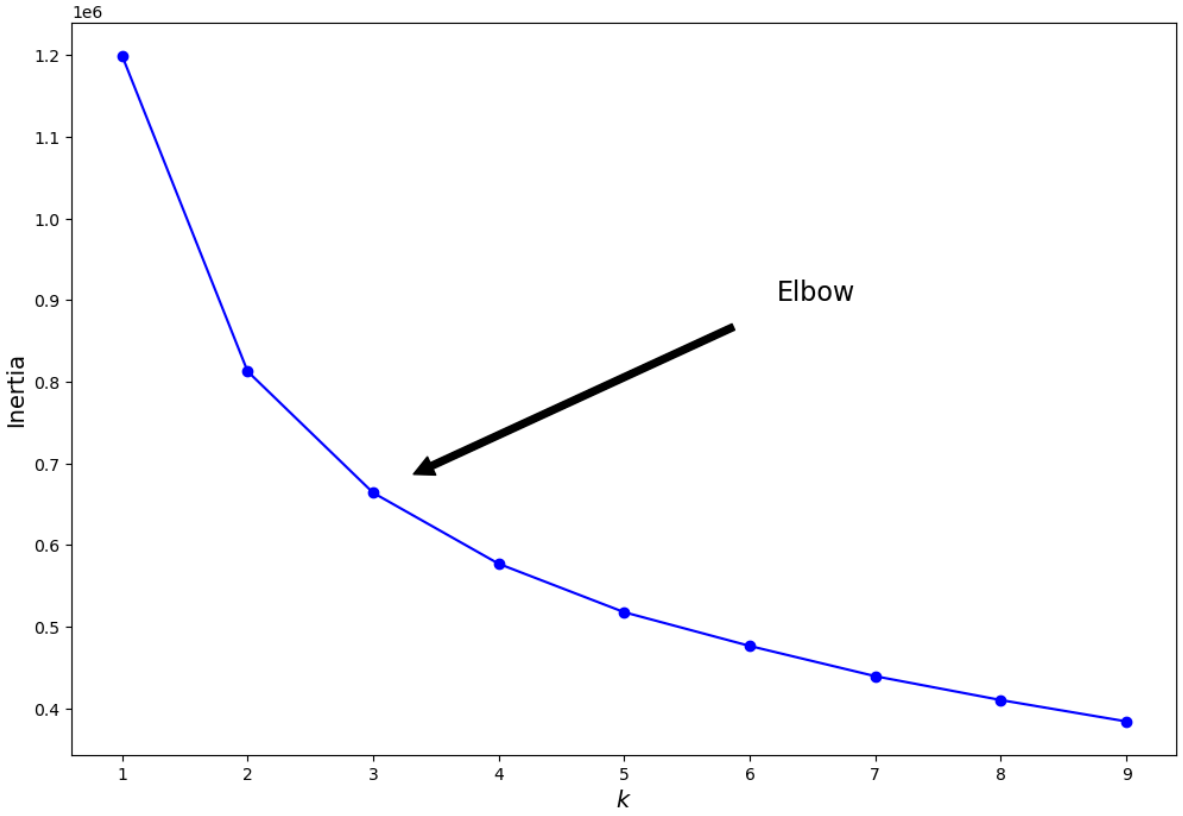
In [107…
```
for i in range(1,10):
 from sklearn.cluster import KMeans
 k = 4
 kM = KMeans(n_clusters=k,
 random_state=654)
 y_pred = kM.fit_predict(X_sc)
```

In [108…
```
kmeans_per_k = [KMeans(n_clusters=k, random_state=42).fit(X_sc)
 for k in range(1, 10)]
inertias = [model.inertia_ for model in kmeans_per_k]
inertias
```

Out[108]:
```
[1199211.9999999984,
 812618.2236265251,
 663951.3689564554,
 577020.6292578045,
 517714.40602218884,
 476402.9017863567,
 439357.96141058847,
 410144.07302076987,
 383988.5907258125]
```

In [109…
```
plt.figure(figsize=(12, 8))
plt.plot(range(1, 10), inertias, "bo-")
plt.xlabel("$k$", fontsize=14)
plt.ylabel("Inertia", fontsize=14)
plt.annotate('Elbow',
 xy=(3, inertias[2]),
 xytext=(0.55, 0.55),
 textcoords='figure fraction',
```

```
 fontsize=16,
 arrowprops=dict(facecolor='black', shrink=0.1)
 )
plt.show()
```



KMeans with n_clusters = 3

In [110…
```
from sklearn.cluster import KMeans
k = 3
kM = KMeans(n_clusters=k,
 random_state=654)
y_pred = kM.fit_predict(X_sc)
```

In [111…
```
clusters = pd.DataFrame(X, columns=X.columns)
clusters['label'] = kM.labels_
```

In [113…
```
clusters.head()
```

Out[113]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience_
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	
2	Others	2015.0	2000000	backend engineer	2020.0	
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	

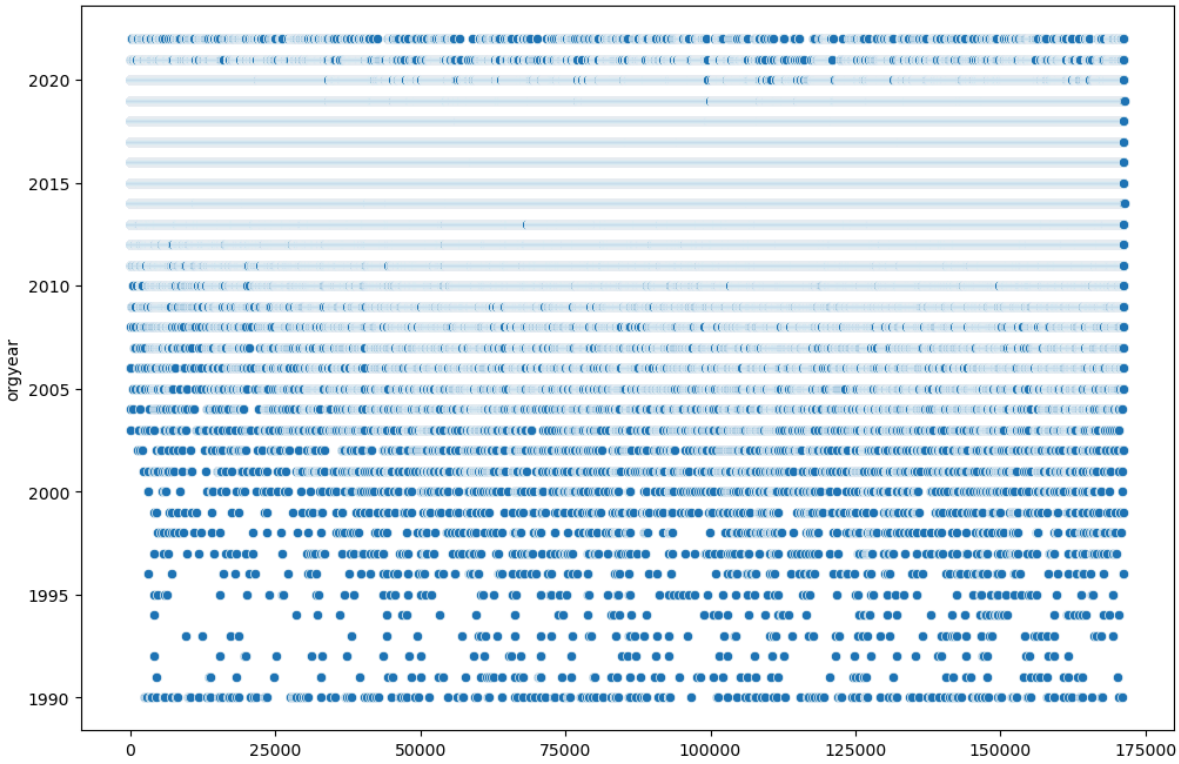
In [114…
```
clusters.shape
```

Out[114]: (171316, 10)

Insights | EDA after Clustering :

In [122…
```
sns.scatterplot(clusters["orgyear"])
```

Out[122]: <AxesSubplot:ylabel='orgyear'>



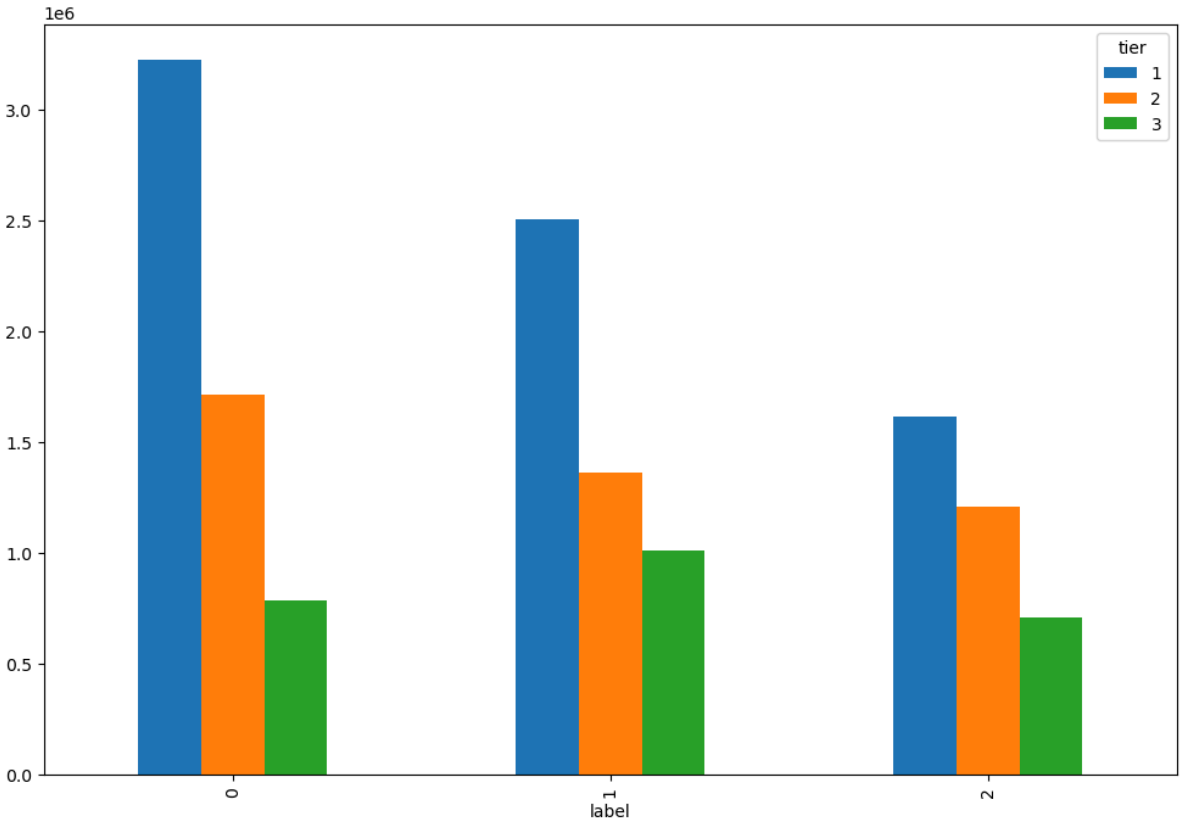
Based on above scatter plot , we can observe , a cluster of learners received CTC upto 30 LPA who joined after 2006-07

There's a group of learners who are very much experienced

And also learners joined after 2012-13 receiving CTC between 20 LPA to upto 1.5cr

In [123…
```
pd.crosstab(index = clusters["label"],
 columns = clusters["tier"],values=clusters["ctc"],aggfunc= np.mean
 ).plot(kind = "bar")
```

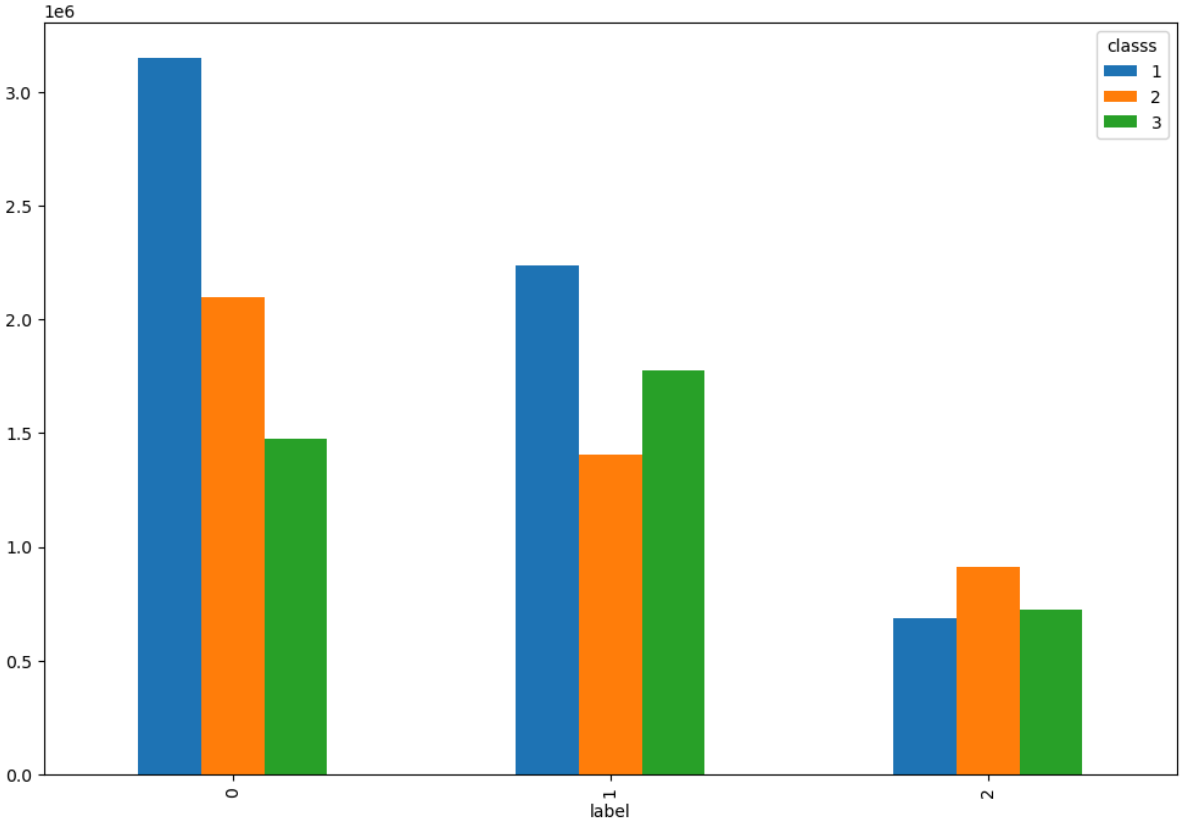
Out[123]: <AxesSubplot:xlabel='label'>



Based on k-Means Clustering algorithm output , as well as manual clustering , learners from tier1 company receiving very high CTC.

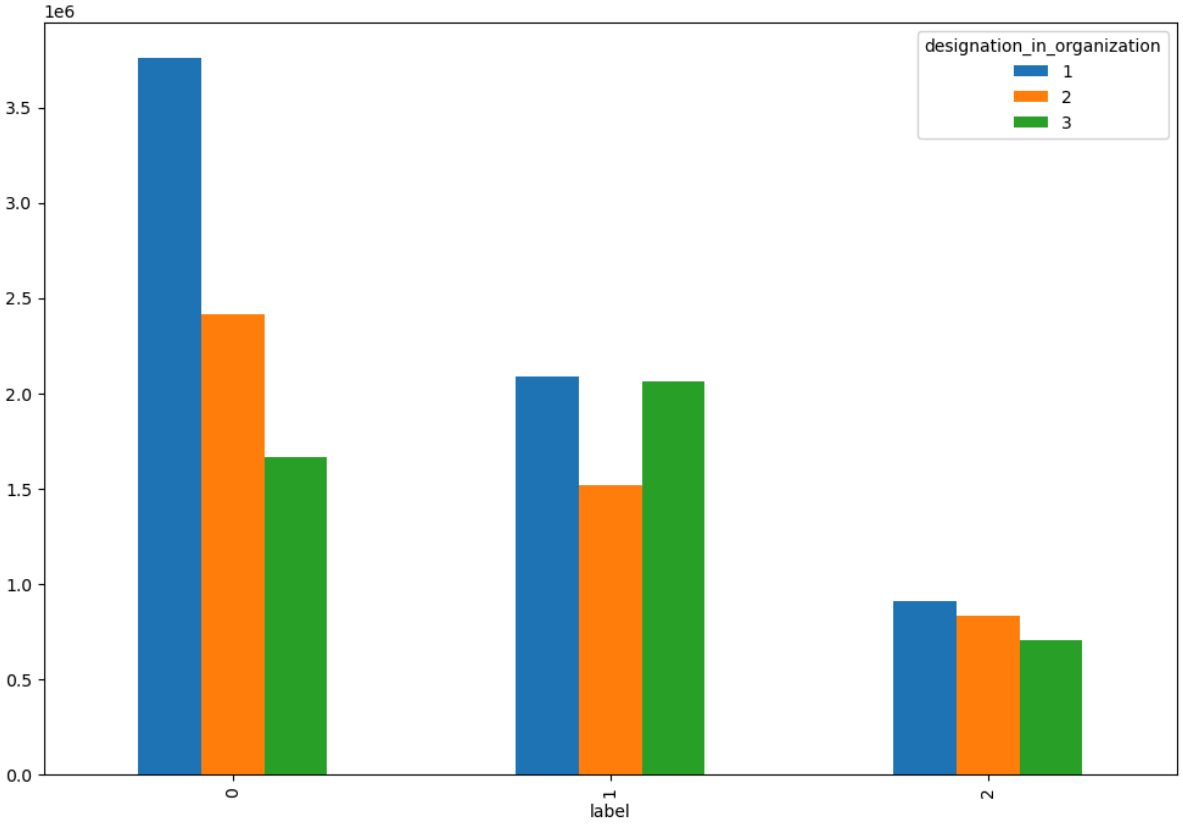
In [124…
```
pd.crosstab(index = clusters["label"],
 columns = clusters["classs"],values=clusters["ctc"],aggfunc= np.mean
 ).plot(kind = "bar")
```

Out[124]:
```
<AxesSubplot:xlabel='label'>
```



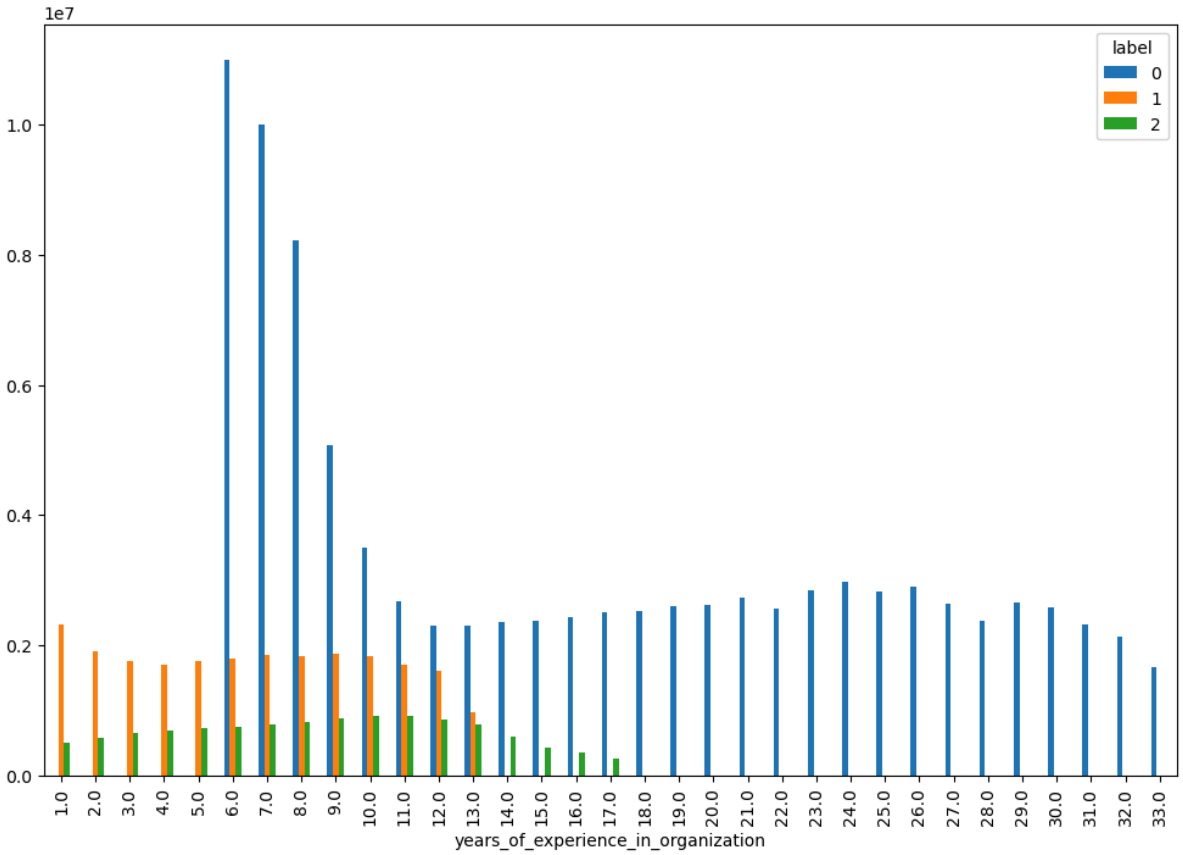
In [125…
```
pd.crosstab(index = clusters["label"],
 columns = clusters["designation_in_organization"],
 values=clusters["ctc"],aggfunc= np.mean
 ).plot(kind = "bar")
```

Out[125]:
```
<AxesSubplot:xlabel='label'>
```



In [126…
```
pd.crosstab(columns = clusters["label"],
 index = clusters["years_of_experience_in_organization"],
 values=clusters["ctc"],aggfunc= np.mean
 ).plot(kind = "bar")
```

Out[126]:
```
<AxesSubplot:xlabel='years_of_experience_in_organization'>
```



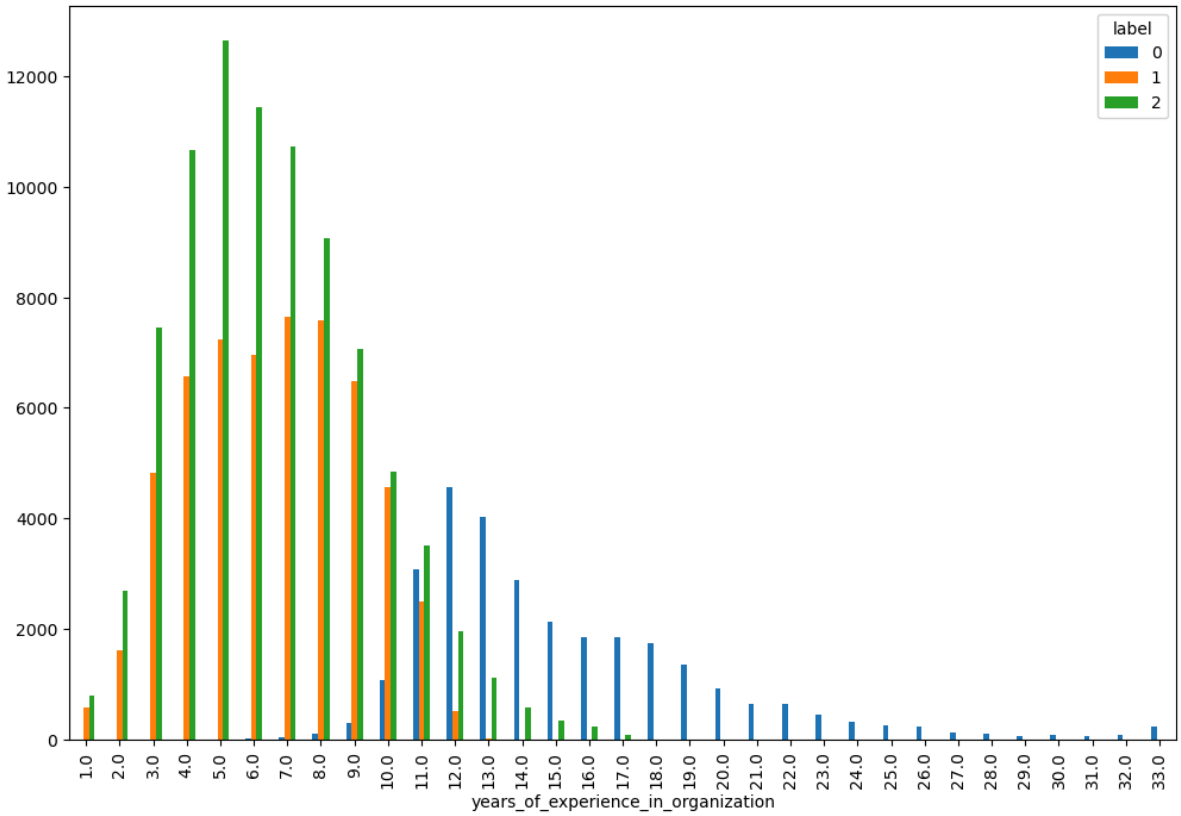
Cluster label 0 , are those learners who are very very experienced,

experienced learners between 6 to 10 years of experience, earning above 40 LPA up tp 1.5Cr.

In [127…
```
pd.crosstab(columns = clusters["label"],
 index = clusters["years_of_experience_in_organization"],

 ).plot(kind = "bar")
```

Out[127]:
```
<AxesSubplot:xlabel='years_of_experience_in_organization'>
```



Majority of Learners are experienced between 1 to 15 years . (49.73%)- (Cluster 2)

there is a group of learners having 8 to upto 33 years of experience. (33%) - (Cluster 0)

16.95% of learners who have experiences - (cluster 1)

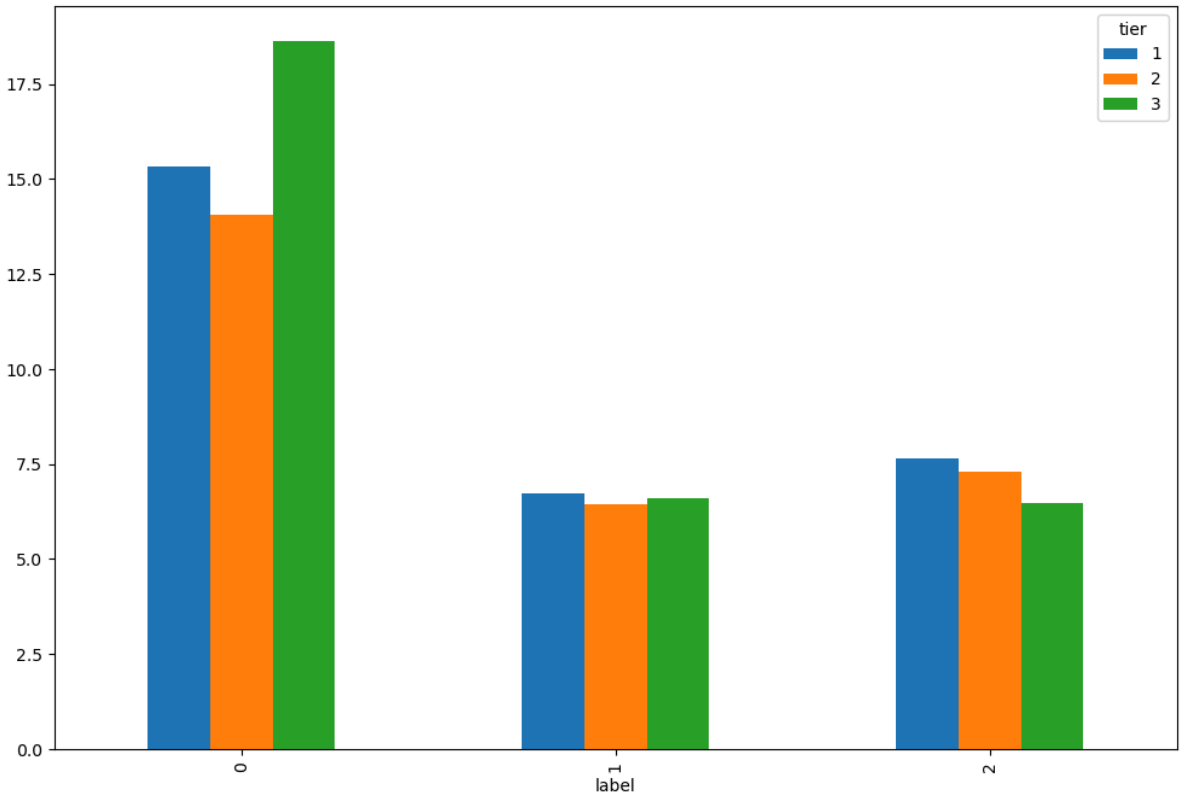
In [128…
```
clusters.label.value_counts(normalize=True)*100
```

Out[128]:
```
2    49.734409
1    33.308623
0    16.956968
Name: label, dtype: float64
```

years_of_experience_in_organization per each cluster group of learners

In [129…
```
pd.crosstab(index = clusters["label"],
 columns = clusters["tier"],
 values=clusters["years_of_experience_in_organization"],
 aggfunc=np.mean
 ).plot(kind = "bar")
```

Out[129]:
```
<AxesSubplot:xlabel='label'>
```



In [130…
```
clusters.columns
```

Out[130]:
```
Index(['company_hash', 'orgyear', 'ctc', 'job_position', 'ctc_updated_yea
r',
       'years_of_experience_in_organization', 'designation_in_organizatio
n',
       'classs', 'tier', 'label'],
      dtype='object')
```

Statistical Summury based on Each Cluster :

In [131…
```
clusters.groupby("label").describe()[["ctc","classs","tier","years_of_exper
```

Out[131]:

	label	0	1	2
ctc	count	2.905000e+04	5.706300e+04	8.520300e+04
	mean	2.543348e+06	1.802940e+06	7.562107e+05
	std	1.751976e+06	1.272597e+06	5.033019e+05
	min	3.955000e+04	6.500000e+04	3.800000e+04
	25%	1.420000e+06	1.000000e+06	4.000000e+05
	50%	2.100000e+06	1.500000e+06	6.300000e+05
	75%	3.147500e+06	2.200000e+06	1.000000e+06
	max	1.250000e+07	1.250000e+07	5.600000e+06
classs	count	2.905000e+04	5.706300e+04	8.520300e+04
	mean	1.625886e+00	1.544574e+00	2.831191e+00
	std	6.937293e-01	5.252113e-01	3.751798e-01
	min	1.000000e+00	1.000000e+00	1.000000e+00
	25%	1.000000e+00	1.000000e+00	3.000000e+00
	50%	2.000000e+00	2.000000e+00	3.000000e+00
	75%	2.000000e+00	2.000000e+00	3.000000e+00
	max	3.000000e+00	3.000000e+00	3.000000e+00
tier	count	2.905000e+04	5.706300e+04	8.520300e+04
	mean	1.484200e+00	1.648774e+00	2.900731e+00
	std	6.478262e-01	5.742163e-01	3.010974e-01
	min	1.000000e+00	1.000000e+00	1.000000e+00
	25%	1.000000e+00	1.000000e+00	3.000000e+00
	50%	1.000000e+00	2.000000e+00	3.000000e+00
	75%	2.000000e+00	2.000000e+00	3.000000e+00
	max	3.000000e+00	3.000000e+00	3.000000e+00
years_of_experience_in_organization	count	2.905000e+04	5.706300e+04	8.520300e+04
	mean	1.520678e+01	6.557945e+00	6.541436e+00
	std	4.339403e+00	2.474935e+00	2.775220e+00
	min	6.000000e+00	1.000000e+00	1.000000e+00
	25%	1.200000e+01	5.000000e+00	4.000000e+00
	50%	1.400000e+01	7.000000e+00	6.000000e+00
	75%	1.700000e+01	8.000000e+00	8.000000e+00
	max	3.300000e+01	1.300000e+01	1.700000e+01

In [139…

```
# Top 10 employees (earning more than most of the employees in the company)
clusters[clusters['tier'] == 1].sort_values('tier',ascending=False).head(10)
```

Out[139]:

	ctc	tier
2	2000000	1
121657	3700000	1
121595	3250000	1
121601	2900000	1
121622	2250000	1
121631	1774000	1
121645	1700000	1
121651	1570000	1
121654	1600000	1
121661	3855000	1

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

In [145…
```
clusters[clusters['tier'] == 3].sort_values('tier',ascending=True).head(10)
```

Out[145]:

	ctc	tier
1	449999	3
110285	1850000	3
110283	900000	3
110282	660000	3
110280	860000	3
110279	710000	3
110278	750000	3
110276	760000	3
110288	750000	3
110275	60000	3

Top 10 companies (based on their CTC)

In [148…
```
df.groupby('company_hash').mean()['ctc'].reset_index().sort_values('ctc',asc
```

Out[148]:

	company_hash	ctc
362	bxwqgonqvntsj	6.337000e+06
3096	wvqttb	6.060375e+06
2814	vxqugqno vhnygqxnj ge xzaxv	4.742857e+06
1551	orxwt	4.548000e+06
592	evzvnxwo xzw	4.383000e+06
2711	vruyvsqtu otwhqxnxto	4.009091e+06
724	gqvwrt wrgha xzeqvonqhwnhqt	3.971667e+06
1207	nqvexshqv	3.960000e+06
163	bgngqi	3.950000e+06
1343	nxat	3.892000e+06

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

In [154…

```
tmp = df[df['job_position'] != 'na']
tmp = tmp.groupby(['company_hash','job_position']).mean().sort_values(['comp
tmp = tmp.groupby('company_hash').head(2)[['company_hash','job_position']]
tmp
```

Out[154]:

	company_hash	job_position
0	Others	research assistant
1	Others	researcher
269	a ntwyzgrgsxto	fullstack engineer
270	a ntwyzgrgsxto	frontend engineer
275	aaqxctz avnv owxtzwto vzvrjnxwo ucn rna	engineering intern
…	…	…
25563	zxyxrtzn ntwyzgrgsxto	android engineer
25569	zxzlvwvqn	other
25570	zxzlvwvqn	area operations manager
25578	zxztrtvuo	other
25579	zxztrtvuo	engineering intern

7414 rows × 2 columns

Insights

- Top Paying job titles are 'Engineering Leadership', 'Backend Engineer', 'Product Manager', 'Program Manager', 'SDET', 'QA Engineer', 'Data Scientist', 'Android Engineer' and 'FullStack Engineer'
- Top paying companies are 'Cisco', 'Intel Technology India Pvt Ltd', 'Amazon', 'Walmart Labs', 'Symantec', 'Schneider Electric India', 'Morgan Stanley', 'Ericsson RD Bangalore' and 'Samsung Electronics'
- Avg CTC seems to be decreasing with year

- Among the top paying companies, salary for these companies are getting lower in recent years, Goldmaan Sachs, Tata Consultancy Services, Samsung Electronics, VMware, Dell, Dbs Bank, Hsbc software devlopement India and GE
- Among Top paying companies, mean salary for these companies are increasing every year, Amazon,Microsoft and Huawei Technologies

Recommendations

- Freshers who want to work on technical side should look for roles related to Backend Engineer, SDET, QA engineer, Data Scientist, Android Engineer,Full stack engineer to get good salaries as expirience increases.
- Freshers who want best CTC should aim for companies like 'Cisco', 'Intel Technology India Pvt Ltd', 'Amazon', 'Walmart Labs', 'Symantec', 'Schneider Electric India', 'Morgan Stanley', 'Ericsson RD Bangalore' and 'Samsung Electronics'.

In [ ]: