Problem Statement: To cluster learners of Scaler based on the company,CTC and experience

In [ ]:
```
Actionable Insights -
1.  There are a total of 37299 unique companies' data
2.  8337 times 'nvnv wgzohrnvzwj otqcxwto' has occurred
3. Most frequent position is a backend engineer
4. CTC updated years are from 2016 to 2021
5. CTC updated for most of the learners in 2019,2020 and 2021
6. The reason for the same email id is the learned could have shifted 2-3 companies in the past. 'CTC-min' gives the
salary of the first company and 'CTC-max' gives the current company salary. The current company's salary is always greater then
 first company's salary
7. Hopkins test score is 0.0007 which is very very close 0 there is a strong tendency for cluster formation and these
points are not from random distributions
8. 12986 data points were removed as they were outliers (7% of data)
9. There are more learners with experience between 5-10 and 0-5 years
10. There are more learners with CTC 5Lto 10L and 10L to 20L
11. Top 10 companies with highest mean CTC is shown in cell no 59 and 138
12. Manual cluster based on company CTC : Tier1-53829, Tier2- 48633, Tier3-90362
13. Cell no 67 to 95 has all the answers to manual clustering
14.CTC and Orgyear have a negative correlation of 0.36
15. CTC and Orgyear have a positive correlation of 0.29
16. Based on elbow method and highest Silhouette avg score the number of clusters are 4
17. Silhouette_avg score is 0.3591
18. Based on hierarchical clustering
Tier 1: mean CTC - 1.327912e+08
Tier 2 : mean CTC - 1.685100e+06
Tier 3 : mean CTC - 1.080494e+06

Recommendations :

1. Use a random forest classifier for imputing missing positions than decisions trees
2. Learners having CTC <10L and <20L are out prime target
3. Learners having <5 years of experience can have better growth than person with more than 15 years of experience
4.  Scaler can build a recommendation system to suggest next top 5 similar positions that learner will land in next 1 year
5. It is good to have learner's Btech degree and year of graduation added to the data
6. Once the learners are placed it is good to add the placed company and CTC in the list for future predictions
7. Some promotional offers to learners who are in the same domain and willing to upskill
8.  Discuss the syllabus in detail with learners during 1:1 discussions before onboarding
9. Build a ML model to give the predicted salary after the course completion
10. Can have some soft skills training, resume building classes
11. Can arrange 1:1 discussions with other learners who are similar to them and have similar background
12. Promotional offers if students join as a group
13. From Time to time taking feedback from learners and taking appropriate action
```

In [1]:
```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings("ignore")
from scipy import stats
from sklearn import preprocessing
import matplotlib.pyplot as plt
import seaborn as sns
import re
from sklearn.manifold import TSNE
from sklearn.preprocessing import StandardScaler
from category_encoders import TargetEncoder
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
from sklearn.manifold import TSNE
from pyclustertend import hopkins
from sklearn.decomposition import PCA
from sklearn.neighbors import LocalOutlierFactor
from sklearn.metrics import silhouette_samples, silhouette_score
```

In [2]:
```
df=pd.read_csv('scaler_clustering.csv')
```

In [3]:
```
df.head(5)
```

Out[3]:

	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

In [4]:
```
df.drop('Unnamed: 0',axis=1,inplace=True)
# Dropping the column
```

In [5]:
```
df.drop_duplicates(keep='first', inplace=True,ignore_index=True)
# Dropping duplicates if any
```

In [6]:
```
df
```

Out[6]:

	company_hash	email_hash	orgyear	ctc	job_position	ctc_updated_year
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	1100000	Other	2020.0
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018.0	449999	FullStack Engineer	2019.0
2	ojzwnvwnxw vx	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2015.0	2000000	Backend Engineer	2020.0
3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	2017.0	700000	Backend Engineer	2019.0
4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	2017.0	1400000	FullStack Engineer	2019.0
...	...	...	...	...	...	...
205805	vuurt xzw	70027b728c8ee901fe979533ed94ffda97be08fc23f33b...	2008.0	220000	NaN	2019.0
205806	husqvawgb	7f7292ffad724ebbe9ca860f515245368d714c84705b42...	2017.0	500000	NaN	2020.0
205807	vwwgrxnt	cb25cc7304e9a24facda7f5567c7922ffc48e3d5d6018c...	2021.0	700000	NaN	2021.0
205808	zgn vuurxwvmrt	fb46a1a2752f5f652ce634f6178d0578ef6995ee59f6c8...	2019.0	5100000	NaN	2019.0
205809	bgqsvz onvzrtj	0bcfc1d05f2e8dc4147743a1313aa70a119b41b30d4a1f...	2014.0	1240000	NaN	2016.0

205810 rows × 6 columns

In [7]:
```
df.info()
# There are null values in company_hash, orgyear and job_position
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205810 entries, 0 to 205809
Data columns (total 6 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   company_hash      205766 non-null  object 
 1   email_hash        205810 non-null  object 
 2   orgyear           205724 non-null  float64
 3   ctc               205810 non-null  int64  
 4   job_position      153263 non-null  object 
 5   ctc_updated_year  205810 non-null  float64
dtypes: float64(2), int64(1), object(3)
memory usage: 9.4+ MB
```

In [8]:
```
df.isnull().sum()
# Count of null values in each column
```

Out[8]:
```
company_hash           44
email_hash              0
orgyear                86
ctc                     0
job_position        52547
ctc_updated_year        0
dtype: int64
```

In [9]:
```
df.columns=['Company','Email','Orgyear','CTC','Position','CTC_updated_year']
# Updating the column names
```

In [10]:
```
df.describe()
# There are incorrect values in Orgyear which needs to be corrected later
# There are outliers in CTC as max CTC >>> median CTC
```

Out[10]:

	Orgyear	CTC	CTC_updated_year
count	205724.000000	2.058100e+05	205810.000000
mean	2014.882284	2.271854e+06	2019.628279
std	63.576199	1.180185e+07	1.325188
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

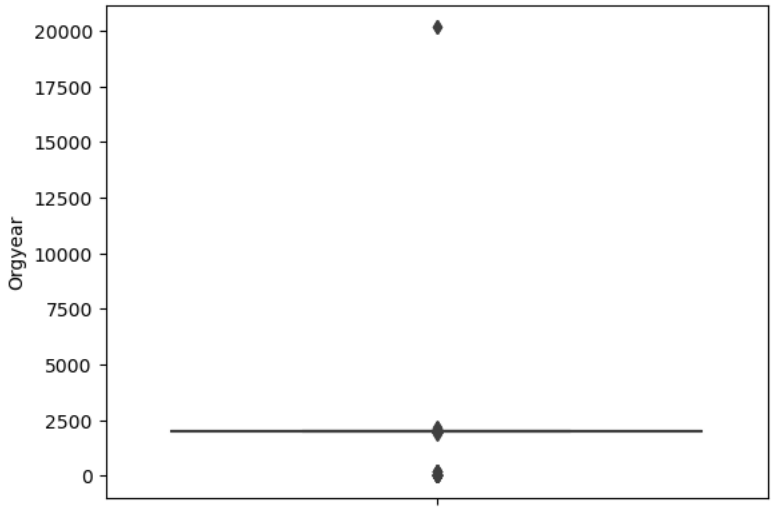
In [11]:
```
df.describe(include=object)
# There are a total of 37299 unique companies data
# 8337 times 'nvnv wgzohrnvzwj otqcxwto' has occured
# Most frequent position is backend engineer
```

Out[11]:

	Company	Email	Position
count	205766	205810	153263
unique	37299	153443	1017
top	nvnv wgzohrnvzwj otqcxwto	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	Backend Engineer
freq	8337	10	43546

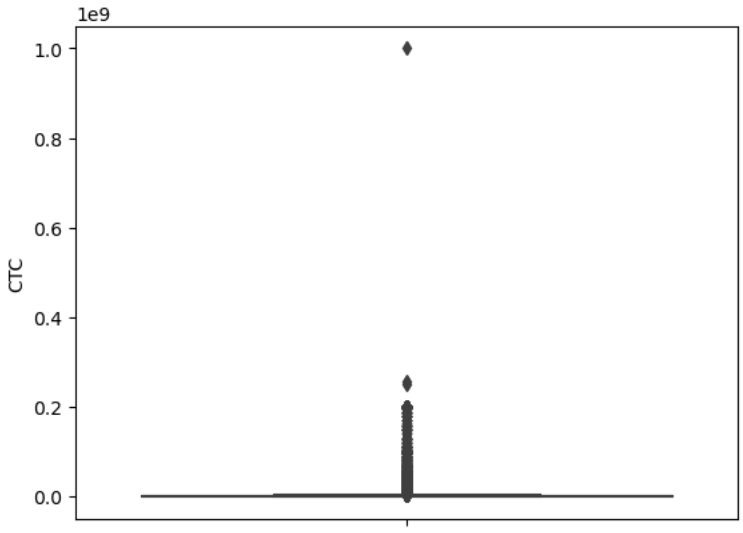
In [12]:
```
sns.boxplot(data=df, y='Orgyear')
# Boxplot below shows that there are outliers in the data in orgyear
```

Out[12]: <AxesSubplot:ylabel='Orgyear'>



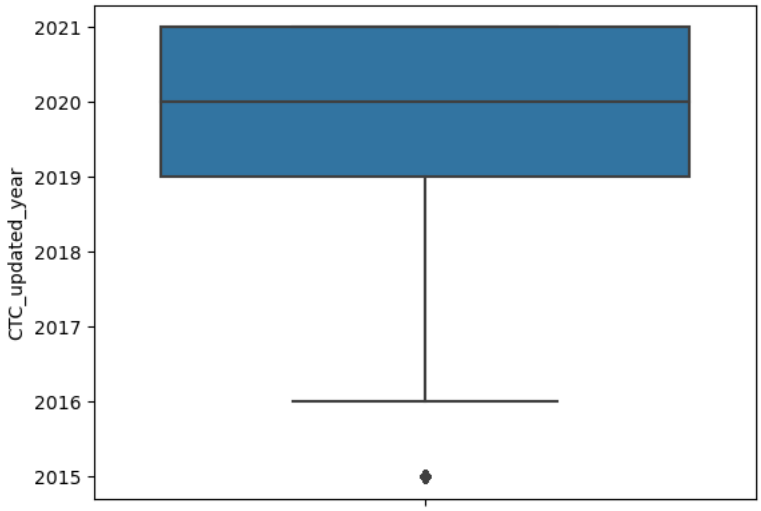
In [13]:
```
sns.boxplot(data=df, y='CTC')
# Boxplot below shows that there are outliers in the data in CTC
```

Out[13]: <AxesSubplot:ylabel='CTC'>



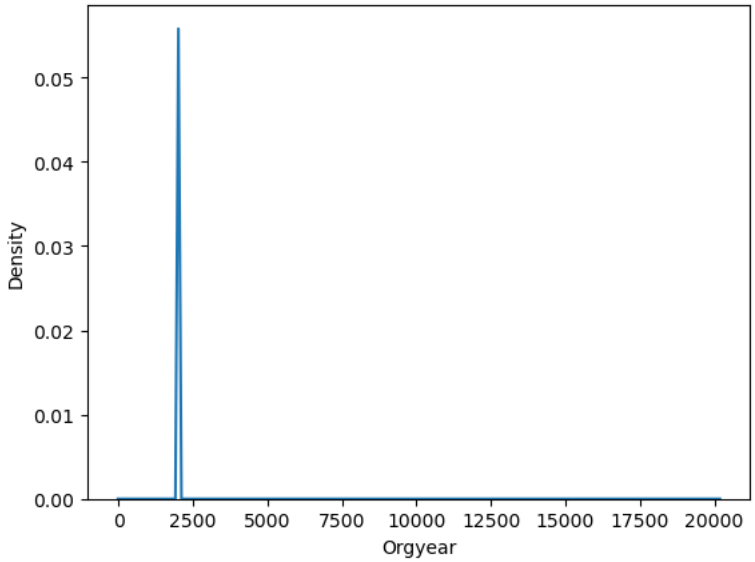
In [14]:
```
sns.boxplot(data=df, y='CTC_updated_year')
# Boxplot below shows that most of the values are between 2016 to 2021
```

Out[14]: <AxesSubplot:ylabel='CTC_updated_year'>



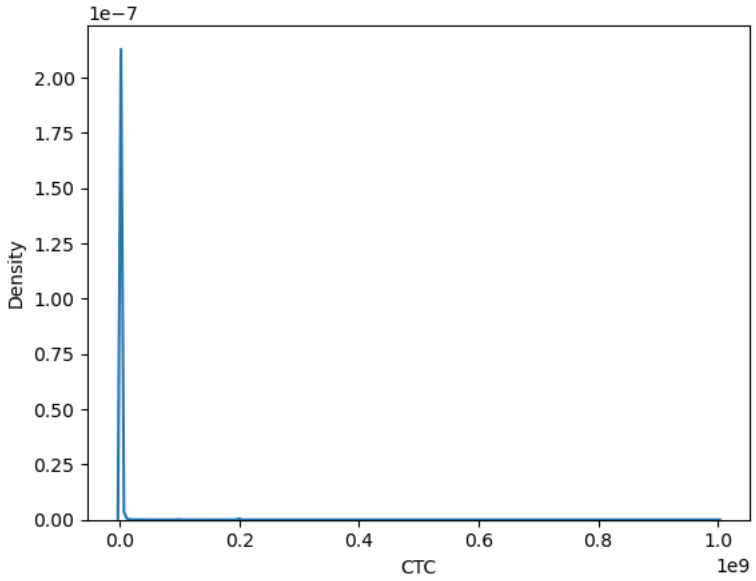
In [15]:
```
sns.kdeplot(data=df, x='Orgyear')
# KDE plot for Orgyear
```

Out[15]: <AxesSubplot:xlabel='Orgyear', ylabel='Density'>



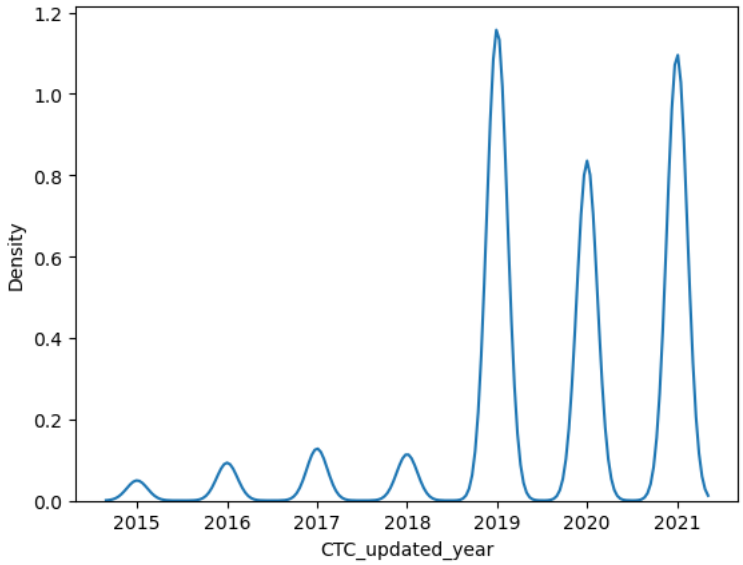
In [16]:
```
sns.kdeplot(data=df, x='CTC')
# KDE plot for CTC
```

Out[16]: <AxesSubplot:xlabel='CTC', ylabel='Density'>



In [17]:
```
sns.kdeplot(data=df, x='CTC_updated_year')
# CTC updated for most of them in 2019,2020 and 2021
```

Out[17]: <AxesSubplot:xlabel='CTC_updated_year', ylabel='Density'>



In [18]:
```
new=df.groupby('Email')[['Email','CTC']].agg({'Email':['count'],'CTC':['min','max','mean']}).reset_index()
```

In [19]:
```
new.columns=['Email','Count','CTC-min','CTC-max','CTC-mean']
```

Groupby Email

In [20]:
```
new[new['Count']>2].sort_values(by=['Count'], axis=0, ascending=False, inplace=False, kind='quicksort', na_position='last')
# Below data frame shows the number of times each email id has occured
# The reason for the same email id is the learned could have shifted 2-3 compaines in the past. 'CTC-min' gives the
# salary of the first company and 'CTC-max' gives the current company salary. Current company's salary is always greater then
# first company's salary
```

Out[20]:

	Email	Count	CTC-min	CTC-max	CTC-mean
112448	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	10	660000	720000	6.900000e+05
25025	298528ce3160cc761e4dc37a07337ee2e0589df251d736...	9	700000	720000	7.111111e+05
37403	3e5e49daa5527a6d5a33599b238bf9bf31e85b9efa9a94...	9	1085000	1130000	1.110000e+06
62459	6842660273f70e9aa239026ba33bfe82275d6ab0d20124...	9	2000000	2400000	2.222222e+06
43145	4818edfd67ed8563dde5d083306485d91d19f4f1c95d19...	8	1200000	1200000	1.200000e+06
...	...	...	...	...	...
58303	6146db59f05cc6116cc7782926a4d85ac46f2fb3aa6223...	3	250000	250000	2.500000e+05
58298	6146233c064fc35ca74888ebdb08ab1fcfd3d8ff26fd5b...	3	300000	370000	3.233333e+05
58287	613e0435d310fa234f9d78118870f136264eaf2cb2a07f...	3	500000	2500000	1.833333e+06
58279	613ac910491f8cb85c8b16119ad89d70c593ae96105a3b...	3	4000000	5300000	4.433333e+06
77846	8223bd059ec36a5599605020373ea19049cddbb3f4595a...	3	340000	440000	4.066667e+05

8655 rows × 5 columns

Missing year and company imputation

In [21]:
```
df['diff']=df['CTC_updated_year']-df['Orgyear']
year_diff=df['diff'].value_counts().index[0]
# Saving the most occured year difference between 'CTC-updated_year' and 'Orgyear'. Using this to calculate the missing Orgyear
company_repe=df['Company'].value_counts().index[0]
# Saving the most occured company. Using this to calculate the missing company as there only 44 missing values
df.drop('diff',axis=1,inplace=True)
```

In [22]:
```
for i in range(len(df)):
    if pd.isna(df.iloc[i]['Company']):
        df.iloc[i, df.columns.get_loc('Company')]=company_repe
    if df.iloc[i]['Orgyear']>=2023:
        df.iloc[i, df.columns.get_loc('Orgyear')]=df.iloc[i]['CTC_updated_year']-year_diff
    elif df.iloc[i]['Orgyear']<1970:
        df.iloc[i, df.columns.get_loc('Orgyear')]=df.iloc[i]['CTC_updated_year']-year_diff
    elif pd.isna(df.iloc[i]['Orgyear']):
        df.iloc[i, df.columns.get_loc('Orgyear')]=df.iloc[i]['CTC_updated_year']-year_diff
# Correcting incorrect values in Orgyear
```

In [23]:
```
df.isnull().sum()
# Now there are only missing values in Position
```

Out[23]:
```
Company                0
Email                  0
Orgyear                0
CTC                    0
Position           52547
CTC_updated_year       0
dtype: int64
```

In [24]:
```
df['Position'].fillna("Unknown", inplace = True)
df['Position']=df['Position'].apply(lambda x: x.lower())
# Filling Nan with Unknown and converting all positions to lower case
```

In [25]:
```
df['Position'].value_counts()
# Value counts of all the positions
```

Out[25]:
```
unknown                 52547
backend engineer        43546
fullstack engineer      25976
other                   18071
frontend engineer       10417
                        ...
third party provider        1
web designer                1
compliance auditor          1
91                          1
azure data factory          1
Name: Position, Length: 945, dtype: int64
```

In [26]:
```
df['Position'].nunique()
# 945 unique positions
```

Out[26]:
```
945
```

Grouping all the positions with similar names

In [27]:
```
dict_roles={'SDE1':['software','software  engineer','Software / web developer','software / web developer''software dev engineer 1
                    'software dev. engineer','software developer (automation)','software developer - android',
 'software developer - ios',
 'software developer - pl10',
 'software developer - ui','software developer 1','software developer consultant','software developer grade-1','software develope
 'software developer/application developer',
 'software development ',
 'software development analyst',
 'software development consultant 1','software development engineer - i', 'software development engineer 1',
 'software development engineer 1 ','software development enginner', 'software devloper',
 'software eng',
 'software engineer  - programmer analyst',
 'software engineer (android)','software development specialist',
 'software engineer - i','software engineer 1','software engineer android',
 'software engineer associate',
 'software engineer i','software engineer in test','software engineer r&d','software engineer, associate',
'software engineering','software engineering contractor','software enginner'
 ,'sde 1', 'software  engineer',
 'software development engineer - i','sde-1',
                    'software developer 1','android application developer','software enginner','sde 1 '
                   ,'sde-i','sde1','Dot net developer','Engineer i'
],

            'SDE2':['software developer 2','software developer ii','software development engineer - 2',
                    'software development engineer - ii','software development engineer -2 ',
 'software development engineer 2',
 'software development engineer 2 ','software development engineer ||','software engineer - 2','software engineer - ii','software
 'software engineer 2 (backend)',
 'software engineer 2 (full stack)',
 'software engineer 2b','software engineer ii','software engineer iii ( sde2) ','software engineer l2',
                    'software engineering - engineer 2',
                    'sde 2','software engineer 2','sde-2','sde2',
                    'software engineer 2 (backend)','software development engineer 2 ',
 'software engineer - ii',
 'software development engineer - 2','sde - ii','sde ii',
'sde-ii',
'sde2',
'sdeii',
'sdet-2',' sde 2','Se ii'],
            'SDE3':['software developer (sde-3)','software development engineer - iii','software development engineer 3',
 'software development engineer iii','software engineer 3','software engineer iii','senior  software engineer',
                    'senior software developer','sde 3','senior software  development engineer',
                    'senior software engineeer','senior developer','sde - 3 ','sde-3',
'sde3',
'sdet 3',],
            'SDE4': ['software development engineer iv','software engineer iv',
                     'software engineering lead','sde 4',
'sde4','Se4'],
           'SDE Analyst/Consultant/Test/Op':['software analyst','software consultant','software engineer - testing',
                                             'software engineer - operations','software engineer analyst ',
                                             'software engineering analyst','software enginnering specialist',
                                             'software prod & plat eng analyst','application development analyst'],
           'SDE Data Engineer':['software developer (data engineer)'],
           'SDE Intern': ['software developer intern','software development engineer - intern',
                          'software development engineer intern','software development engineering intern',
                          'software development intern','software engineer (intern)','software engineer android intern',
                          'software engineer intern','software engineer(r&d) intern ',
                          'software engineering intern','intern','engineering intern',
                          'intern - software developer','sde intern','machine learning engineer intern',
 'machine learning intern','mts intern','Sdet - intern'],
           'SDE Manager':['software development manager'],
           'SDE ML':['software engineer trainee(ml)','software engineer(advanced analytics)',
                     'software engineering co-op','machine learning engineer','machine learning data associate ',
 'machine learning developer','ml engineer'],
            'Backend Engineer':['backend engineer','software engineer (backend)','software engineer (backend)','backend engineeri
            'Fullstack Engineer':['fullstack engineer','software engineer (full stack)',
                                  'software engineer (full stack)','full stack engineer','Full stack devloper','full stack devlop
 'full stack web developer',
 'full-stack web developer',
 'fullstack engineer',
 'system engineer,fullstack developer',
 'web developer (full stack)'],
            'Frontend Engineer':['frontend engineer','software engineer - frontend','Front end developer'],
            'Engineering Leadership':['engineering leadership'],
 'QA Engineer':['qa engineer','software qa',
 'software qa engineer'],
 'Data Scientist':['data scientist'],
 'Android Engineer':['android engineer'],
 'SDET':['sdet','software development engineer in test','software test engineer','manual tester',
         'Test automation engineer','Tester','Tester'],
 'Devops Engineer':['devops engineer'],
 'Support Engineer':['support engineer'],
 'Data Analyst':['data analyst'],
 'Ios Engineer':['ios engineer'],

 'Product Designer':['product designer'],
```

```
 'Backend Architect':['backend architect'],
 'Research Engineers':['research engineers'],
 'Product Manager':['product manager'],
 'Program Manager':['program manager'],
 'Non-coder':['non coder'],
 'Database Administrator':['database administrator'],
 'Co-founder':['co-founder'],
 'Security Leadership':['security leadership'],
 'Release Engineer':['release engineer'],
            'System Engineer':['system engineer'],
            'ASE':['associate software engineer','associate','assistant system engineer',
                   'associate  software  engineer ','associate software developer','Ase'],
            'Senoir Engineer/Consultant':['consultant','senior engineer','senior consultant','associate consultant',
                                          'senior systems engineer','senior software engineer (backend)'],
            'Data Engineer':['data eingineer','data engineer'],
            'Data Engineer 2':['data engineer 2'],
            'Data Engineer 3':['data engineer iii'],
            'Data Scientist':['data scientist'],
            'Data Scientist 2':['data scientist 2','data scientist ii'],
            'Unknown':['.', '..','.7','7','737','91','ays','ba','bdm','bta','coo','crc','h',
                       'kam','l4','n','na','nce','no','none','null','owner','pa',
                       'pat','pop','pune','q','so','sr.','sre','storw','telar','tsc','ucm','x','896651','857628',"Can't reveal",
                       'Typing','Bengaluru','7033771951'],
            'Mechnaical Engineer':['mechanical engineering ','mechanical engineers'],
            'Member of technical staff 1':['member of technical staff (java)',
 'member of technical staff (mts)','member of technical staff at nineleaps', 'member technial staff',
 'member technical',
 'member technical staff',
 'member technical staff -1','member technical staff zoho'],
            'Member of technical staff 2':['member of technical staff - 2',
 'member of technical staff 2','member of technical staff level 2','member technical staff 2',],
            'Member of technical staff 3':['member of technical staff 3','member technical staff iii',],
            'Member of technical staff 4':['member of technical staff 4',],
            'MTS 1':['mts'],
            'MTS 2':['mts - ii','mts-2','mts2'],
            'MTS 3':['mts 3','mts-3'],
            'Associate product manager':['assistant manager',
 'assistant manager @ capital market ',
 'assistant regional sales manager',
 'associate manager',
 'associate product manager',
 'associate project manager ',
 'assosiate product manager',],
    'Manager':['manager',
 'manager ',
 'manager candidate',
 'manager-cx',]



          }
```

In [28]:
```
def applying(x,dict_roles):
    for i,j in dict_roles.items():
        if x in j:
            return i
    return x.capitalize()
```

In [29]:
```
df['Position']=df['Position'].apply(lambda x: applying(x,dict_roles))
df['Position']=df['Position'].replace('Unknown', np.NaN)
```

In [30]:
```
df['Position'].nunique()
# Now there are total of 742 values
```

Out[30]:
```
742
```

In [31]:
```
df.isnull().sum()
```

Out[31]:
```
Company                0
Email                  0
Orgyear                0
CTC                    0
Position           52595
CTC_updated_year       0
dtype: int64
```

Checking clustering tendency - Hopkins test

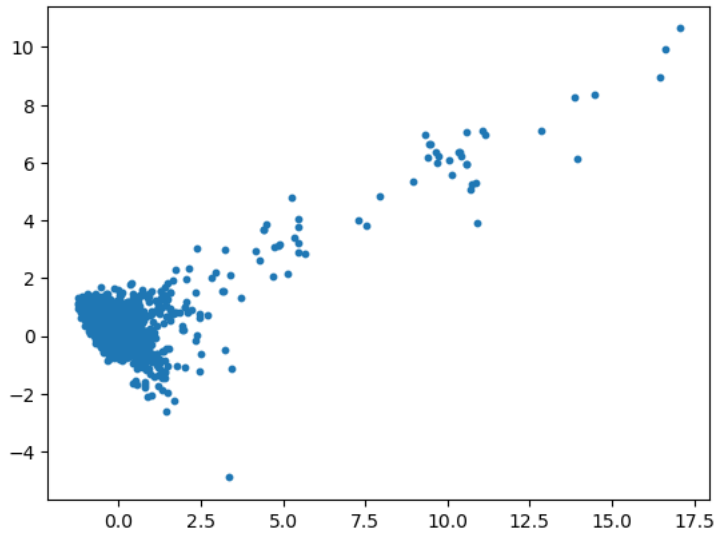
In [32]:
```
df_st=df.copy()
# Statistocal test
df_st.drop('Email',axis=1,inplace=True)
from category_encoders import TargetEncoder
Targetenc = TargetEncoder(cols=['Company','Position'])
df_st = Targetenc.fit_transform(X = df_st, y = df_st['CTC'])
scaler = StandardScaler()
df_st=scaler.fit_transform(df_st)
df_st=pd.DataFrame(df_st)
hopkins(df_st,len(df_st))

# As the value of hopkins test is 0.0007 which is very very close 0 there is strong tendency for cluster formation and these
# points are not from random distributions
```

Out[32]:
```
0.0007349957300649914
```

PCA scatterplot

In [33]:
```
pca=PCA(n_components=2).fit(df_st)
z1=pca.transform(df_st)
fig,ax=plt.subplots()
im=ax.scatter(z1[1000:3000,0],z1[1000:3000,1],s=10)
# We are unable to say if there are different clusters possible
```



Outliner removal- LOR method

In [34]:
```
clf = LocalOutlierFactor(n_neighbors=900)
ar=clf.fit_predict(df_st)
```

In [35]:
```
pd.Series(ar).value_counts()
# There are 12986 outliers in the data set
```

Out[35]:
```
 1    192824
-1     12986
dtype: int64
```

In [36]:
```
df_st['outlier']=pd.Series(ar)
```

In [37]:
```
index_vals=df_st[df_st['outlier']==-1].index
index_vals=list(index_vals)
```

In [38]:
```
df.drop(index_vals,inplace=True)
# removing these data form df data frame
```

Imputing missing positions using decision trees

In [39]:
```
df1=df[df.isnull().any(axis=1)]
df1.reset_index(inplace=True)
df1_email=df1['Email']
df1.drop('index',axis=1,inplace=True)
df1.drop('Email',axis=1,inplace=True)
df2=df.dropna(axis=0,how='any')
df2.reset_index(inplace=True)
df2_email=df2['Email']
df2.drop('index',axis=1,inplace=True)
df2.drop('Email',axis=1,inplace=True)
df.drop('Email',axis=1,inplace=True)
```

In [40]:
```
from category_encoders import TargetEncoder
Targetenc = TargetEncoder(cols=['Company'])
df = Targetenc.fit_transform(X = df, y = df['CTC'])
```

In [41]:
```
df3=df[df.isnull().any(axis=1)]
df3.reset_index(inplace=True)
df3.drop('index',axis=1,inplace=True)
df4=df.dropna(axis=0,how='any')
df4.reset_index(inplace=True)
df4.drop('index',axis=1,inplace=True)
```

Split data into 2 parts. One does not have missing values and other has missing values.

Using Supervised learning by taking 'Position' as y parameter. Then predicting missing values

In [42]:
```
from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier(criterion='log_loss',random_state=0, max_depth=700)
decision_tree.fit(df4.drop('Position',axis=1), df4['Position'])
```

Out[42]:
```
DecisionTreeClassifier(criterion='log_loss', max_depth=700, random_state=0)
```

In [43]:
```
decision_tree.score(df4.drop('Position',axis=1), df4['Position'])
# Able to achieve score of 80% for train data set. We can use other ensemble learning techniques to impute missing values
# As the data size is huge my system is giving memory limit error hence used decision trees
```

Out[43]:
```
0.7959929577464788
```

In [44]:
```
df3['Position']=decision_tree.predict(df3.drop('Position',axis=1))
df3['Company']=df1['Company']
df3['Email']=df1_email
df4['Company']=df2['Company']
df4['Email']=df2_email
```

In [45]:
```
df5=pd.concat([df3,df4],axis=0)
```

In [46]:
```
df5['Experience']=2023-df5['Orgyear']
# Feature engineering for experience
```

In [47]:
```
df_bin=df5.copy()
```

Binning experience and CTC column

In [48]:
```
bins = [0, 5,10,15,20,25,30,35,40]
labels = ['0-5', '5-10', '10-15','15-20','20-25','25-30','30-35','35-40']
df_bin['Exp_bin'] = pd.cut(x = df_bin['Experience'], bins = bins, labels = labels, include_lowest = True)

bins = [0, 500000,1000000,2000000,5000000,10000000,200000001]
labels = ['0-500000', '500000-1000000', '1000000-2000000','2000000-5000000','5000000-10000000','10000000-200000001']
df_bin['CTC_bin'] = pd.cut(x = df_bin['CTC'], bins = bins, labels = labels, include_lowest = True)
```

In [49]:
```
df_bin
```

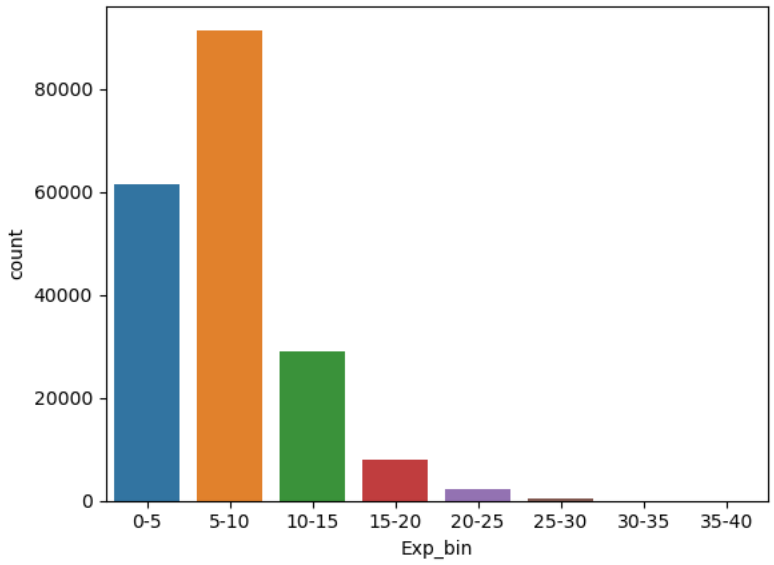
Out[49]:

	Company	Orgyear	CTC	Position	CTC_updated_year	Email	Experience	Exp_bin	CTC_bin
0	utqoxontzn ojontbo	2020.0	450000	Backend Engineer	2019.0	e245da546bf50eba09cb7c9976926bd56557d1ac9a17fb...	3.0	0-5	0-500000
1	xrbhd	2019.0	360000	SDE Intern	2019.0	b2dc928f4c22a9860b4a427efb8ab761e1ce0015fba1a5...	4.0	0-5	0-500000
2	mvqwrvjo wgqugqvnt mvzpxzs	2020.0	800000	Backend Engineer	2020.0	7f24d2f5171ea469482a9966832237bc023678883ecd0c...	3.0	0-5	500000-1000000
3	puxn	2020.0	1400000	Fullstack Engineer	2019.0	26b502eb6439ac80bd618a6f7c2b1c640b84c1e64c472c...	3.0	0-5	1000000-2000000
4	mvlvl exzotqc	2018.0	100000	Other	2021.0	62d2e04b44c8bf2f6ec15d5b4c259c06199f598dc51816...	5.0	0-5	0-500000
...	...	...	...	...	...	...	...	...	...
141995	wos xzntqzvnxgzvr	2016.0	1500000	Fullstack Engineer	2021.0	ee1e251ebb54cffeddc09244bbc0ed122a1cf59511bbc1...	7.0	5-10	1000000-2000000
141996	xzegojo	2019.0	1200000	Fullstack Engineer	2021.0	aec7061c552cfd56fe635ca0e9d347d2e63a95dbcd2ef8...	4.0	0-5	1000000-2000000
141997	wgbuzgcv wgznqvwn	2015.0	1000000	Data Scientist	2021.0	251f5f33672f58c65f43e94d0fc0fef69a7f6149b3b4db...	8.0	5-10	500000-1000000
141998	ahzzyhbmj	2019.0	1100000	Data Scientist	2021.0	617a49afbc5efc0f1692780d20fc566ba4ed4f1125491c...	4.0	0-5	1000000-2000000
141999	ertdnqvat ojontbo rbn	2017.0	1100000	Frontend Engineer	2021.0	8400c092bfe0b5baca38888c8cf2b9fda5e206c9812a22...	6.0	5-10	1000000-2000000

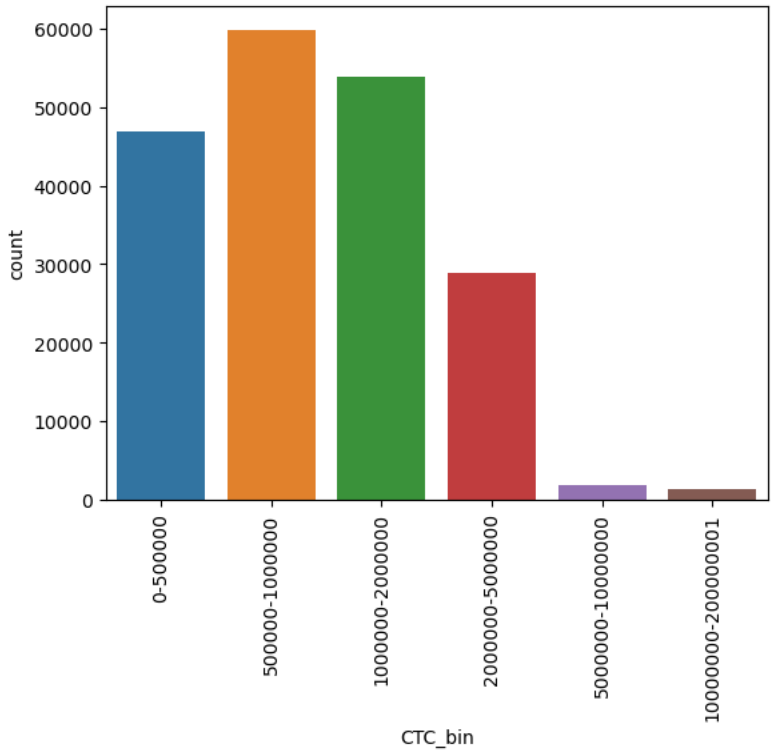
192824 rows × 9 columns

In [50]:
```
sns.countplot(x=df_bin['Exp_bin'])
# There are more learners with experience between 5-10 and 0-5 years
```

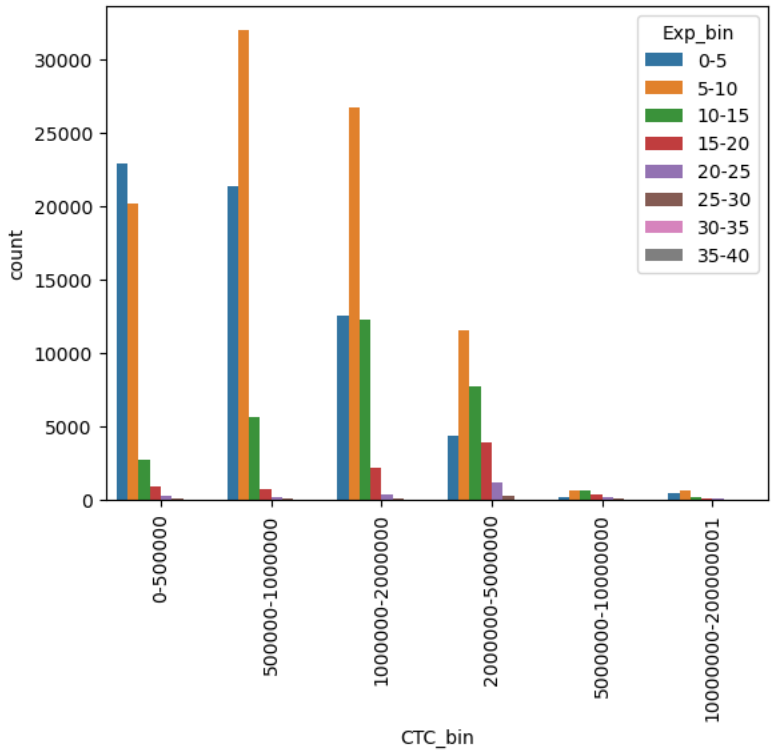
Out[50]: <AxesSubplot:xlabel='Exp_bin', ylabel='count'>



In [51]:
```
sns.countplot(x=df_bin['CTC_bin'])
plt.xticks(rotation=90)
plt.show()
# There are more learners with CTC 5Lto 10L and 10L to 20L
```



In [52]:
```
sns.countplot(x=df_bin['CTC_bin'],hue=df_bin['Exp_bin'])
plt.xticks(rotation=90)
plt.show()
# Count plot for experience vs CTC
```



Regex for position and company

In [53]:
```
def regex(x):
    return(re.sub('[^A-Za-z0-9 ]+', '', x))
```

In [54]:
```
df5['Company']=df5['Company'].apply(lambda x:regex(x) )
df5['Position']=df5['Position'].apply(lambda x:regex(x) )
```

In [55]:
```
orgi=df5.copy('deep')
```

In [56]:
```
df6=df5.groupby(['Company'])[['CTC','Experience']].agg({'CTC':['mean','count','min','max','median'],'Experience':['mean','count',
```

In [57]:
```
df6.columns=['Company',
             'CTC-mean',
             'CTC-count',
                    'CTC-min',
                    'CTC-max',
                    'CTC-median',
             'Experience-mean',
             'Experience-count',
             'Experience-min',
             'Experience-max',
             'Experience-median']
```

In [58]:
```
df6.sort_values(by=['CTC-mean'], axis=0, ascending=False, inplace=True, kind='quicksort', na_position='last')
```

Getting the 5 point summary of CTC (mean, median, max, min, count etc)

Top 10 companies with highest mean CTC and having count >500

In [59]:
```
df6[df6['CTC-count']>500].head(10)
```

Out[59]:

	Company	CTC-mean	CTC-count	CTC-min	CTC-max	CTC-median	Experience-mean	Experience-count	Experience-min	Experience-max	Experience-median
34807	zgzt	6.446076e+06	514	1000	200000000	655000.0	4.571984	514	1.0	12.0	4.0
34718	zgn vuurxwvmrt	5.577645e+06	970	7300	200000000	750000.0	3.877320	970	1.0	22.0	3.0
25563	vbvkgz	3.265368e+06	3245	2000	200000000	2000000.0	7.267180	3245	2.0	27.0	7.0
3665	bxwqgogen	3.071779e+06	1997	5000	200000000	2500000.0	8.753630	1997	2.0	28.0	8.0
12652	ntwy bvyxzaqv	3.070897e+06	773	10000	200000000	600000.0	8.522639	773	1.0	25.0	8.0
5747	erxupvqn	3.020792e+06	920	10000	200000000	2530000.0	7.942391	920	2.0	25.0	7.0
20715	sgrabvz ovwyo	2.941501e+06	759	1000	100000000	2623000.0	7.281950	759	2.0	27.0	7.0
25290	vagmt	2.724995e+06	1314	8000	100010000	2250000.0	8.864536	1314	2.0	24.0	9.0
6786	fvrbvqn rvmo	2.645040e+06	721	13000	10000000	2500000.0	9.128988	721	2.0	23.0	8.0
9803	lubgqsvz wyvot wg	2.605612e+06	1080	11000	200000000	1800000.0	9.866667	1080	1.0	30.0	10.0

Top companies with highest mean CTC and having count >500

In [60]:
```
df6
```

Out[60]:

	Company	CTC-mean	CTC-count	CTC-min	CTC-max	CTC-median	Experience-mean	Experience-count	Experience-min	Experience-max	Experience-median
5920	evwxrt otqcxwto	200000000.0	1	200000000	200000000	200000000.0	5.0	1	5.0	5.0	5.0
35018	ztbyvzo ogrhnxgzo ucn rna	200000000.0	1	200000000	200000000	200000000.0	7.0	1	7.0	7.0	7.0
12858	ntwyzg xzaxv sqghu	200000000.0	1	200000000	200000000	200000000.0	7.0	1	7.0	7.0	7.0
25433	vayxxuv ntwyzgrgsxto ucn rna	200000000.0	1	200000000	200000000	200000000.0	9.0	1	9.0	9.0	9.0
7377	gqmxn ogenfvqt xzw	200000000.0	1	200000000	200000000	200000000.0	7.0	1	7.0	7.0	7.0
...	...	...	...	...	...	...	...	...	...	...	...
23889	uqgmrtb ogrcxzs	500.0	1	500	500	500.0	5.0	1	5.0	5.0	5.0
25711	vcvzn sqghu	300.0	1	300	300	300.0	10.0	1	10.0	10.0	10.0
6457	ftm ongqt	25.0	1	25	25	25.0	7.0	1	7.0	7.0	7.0
24151	uqvpqxnx voogwxvnto	24.0	1	24	24	24.0	3.0	1	3.0	3.0	3.0
31601	xm	15.0	1	15	15	15.0	7.0	1	7.0	7.0	7.0

35934 rows × 11 columns

Manual Clustering based on Company CTC

Analysis at the Company level. Name that flag Tier with values [1,2,3]

In [61]:
```
def q1(x):
    return x.quantile(0.33)
def q3(x):
    return x.quantile(0.66)
vals = {'CTC': [q1, q3]}
df6=df5.groupby('Company').agg(vals).reset_index()
```

In [62]:
```
df6.columns=['Company','CTC-1','CTC-2']
```

In [63]:
```
df_comp=orgi.merge(df6,on='Company')
```

In [64]:
```
df_comp['Tier_company']=0
```

In [65]:
```
for i in range(len(df_comp)):
    if df_comp.iloc[i]['CTC']<=df_comp.iloc[i]['CTC-1']:
        df_comp.iloc[i, df_comp.columns.get_loc('Tier_company')]=3
    elif df_comp.iloc[i]['CTC']<=df_comp.iloc[i]['CTC-2']:
        df_comp.iloc[i, df_comp.columns.get_loc('Tier_company')]=2
    else:
        df_comp.iloc[i, df_comp.columns.get_loc('Tier_company')]=1
```

In [66]:
```
df_comp['Tier_company'].value_counts()
# Below is the count of number of companies in each Tier
```

Out[66]:
```
3    90362
1    53829
2    48633
Name: Tier_company, dtype: int64
```

In [132]:
```
df_comp.groupby('Tier_company')['CTC'].mean()
# Below is the count of number of companies in each Tier based on CTC
```

Out[132]:
```
Tier_company
1    4.027553e+06
2    1.181075e+06
3    1.352364e+06
Name: CTC, dtype: float64
```

Top 10 companies (based on their CTC)

In [138]:
```
df_comp.groupby('Company')['CTC'].mean().reset_index().sort_values('CTC',ascending=False).head(10)
```

Out[138]:

	Company	CTC
5920	evwxrt otqcxwto	200000000.0
35018	ztbyvzo ogrhnxgzo ucn rna	200000000.0
12858	ntwyzg xzaxv sqghu	200000000.0
25433	vayxxuv ntwyzgrgsxto ucn rna	200000000.0
7377	gqmxn ogenfvqt xzw	200000000.0
23706	uhxoovzwt xn vacxogqj vza exzvzwxvr otqcxwto rru	200000000.0
12908	ntwyzguvqp xnqvxnk xn ogrhnxgz	200000000.0
21706	swtpvqva	200000000.0
31402	xbutrojo	200000000.0
4189	ctqexohayv ntwyzgrgsxto	200000000.0

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

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

In [67]:
```
df_co=df_comp[df_comp['Position'].str.startswith('Data ')]
```

Assuming 'nvnv wgzohrnvzwj otqcxwto' as TCS

Assuming 'wgszxkvzn ' as Amazon

Top 10 employees of TCS

In [70]:
```
df_co[(df_co['Company']=='nvnv wgzohrnvzwj otqcxwto') & (df_co['Tier_company']==1)].sort_values('CTC',ascending=False).head(10)
```

Out[70]:

	Company	Orgyear	CTC	Position	CTC_updated_year	Email	Experience	CTC-1	CTC-2	Tier
17834	nvnv wgzohrnvzwj otqcxwto	2015.0	200000000	Data Analyst	2020.0	82fbacc015757ee74efe763c3623dc751812af48c741ff...	8.0	400000.0	620000.0	
21209	nvnv wgzohrnvzwj otqcxwto	2015.0	200000000	Data Analyst	2020.0	82fbacc015757ee74efe763c3623dc751812af48c741ff...	8.0	400000.0	620000.0	
20554	nvnv wgzohrnvzwj otqcxwto	2015.0	200000000	Data Analyst	2020.0	22fa43a2a1bfc28b84acf5b7b6267bb9f8feb76ad3beeb...	8.0	400000.0	620000.0	
21796	nvnv wgzohrnvzwj otqcxwto	2015.0	200000000	Data Analyst	2020.0	1ad9ecf5763e6fcc1e7204355c2645e12d3baaf7a7c083...	8.0	400000.0	620000.0	
21520	nvnv wgzohrnvzwj otqcxwto	2015.0	200000000	Data Analyst	2020.0	59316048d113539202325e05af9b66620255ba84eab635...	8.0	400000.0	620000.0	
21288	nvnv wgzohrnvzwj otqcxwto	2017.0	200000000	Data Analyst	2020.0	655da5cd99f1ba4ad249dade5039b914023484fb7f3959...	6.0	400000.0	620000.0	
21284	nvnv wgzohrnvzwj otqcxwto	2012.0	200000000	Data Analyst	2020.0	d42cf076c3915454a083788ea22df68a65a294563adfa3...	11.0	400000.0	620000.0	
22333	nvnv wgzohrnvzwj otqcxwto	2020.0	100000000	Data Analyst	2020.0	8212e1a25405440bf29178cd87f89bb1ad33108e3cbc04...	3.0	400000.0	620000.0	
24240	nvnv wgzohrnvzwj otqcxwto	2002.0	10000000	Data Analyst	2021.0	c74d843588ff8b4018cfd538dc8f75207a7649e29ba4d6...	21.0	400000.0	620000.0	
19516	nvnv wgzohrnvzwj otqcxwto	2002.0	10000000	Data Analyst	2021.0	c74d843588ff8b4018cfd538dc8f75207a7649e29ba4d6...	21.0	400000.0	620000.0	

Bottom 10 employees of TCS

In [71]:
```
df_co[(df_co['Company']=='wgszxkvzn') & (df_co['Tier_company']==1)].sort_values('CTC',ascending=False).head(10)
```

Out[71]:

	Company	Orgyear	CTC	Position	CTC_updated_year	Email	Experience	CTC-1	CTC-2	Tier_co
13450	wgszxkvzn	2014.0	200000000	Data Analyst	2020.0	aad581a532f319c76c6e73937572feed9867d5ee2f1093...	9.0	450000.0	736400.0	
14371	wgszxkvzn	2021.0	100000000	Data Analyst	2020.0	feaefa9e48e79a7a31d721616f6502cfc59feb72e74f87...	2.0	450000.0	736400.0	
13596	wgszxkvzn	2014.0	100000000	Data Analyst	2020.0	d3782a59939d4cc9e05d35b8de613d0f1a8fcf49d9558b...	9.0	450000.0	736400.0	
12921	wgszxkvzn	2020.0	2300000	Data Scientist	2019.0	cc91430bac8e77dc9df81af64da439b3f9746d37a739dc...	3.0	450000.0	736400.0	
13062	wgszxkvzn	2014.0	2200000	Data Analyst	2021.0	6da72e79a161af77b75aa92c986f5000cc42cf117814d7...	9.0	450000.0	736400.0	
13586	wgszxkvzn	2014.0	2200000	Data Analyst	2021.0	6da72e79a161af77b75aa92c986f5000cc42cf117814d7...	9.0	450000.0	736400.0	
13833	wgszxkvzn	2015.0	1800000	Data Scientist	2021.0	c390ecdc597ea41fb2bbb5988e87d66592f43af76f3870...	8.0	450000.0	736400.0	
15328	wgszxkvzn	2019.0	1700000	Data Analyst	2020.0	d058fb52333639d3ef5ba97a162f2f1d9b1b8c7b78e1e1...	4.0	450000.0	736400.0	
13018	wgszxkvzn	2020.0	1700000	Data Scientist	2019.0	df12fe0032b404e0bbb0f2c28b52b0808349f280ca5a81...	3.0	450000.0	736400.0	
13544	wgszxkvzn	2014.0	1560000	Data Analyst	2020.0	a6303c6c793d81b2cdd51dfc9af375275830124c273f3e...	9.0	450000.0	736400.0	

Top 10 employees of Amazon

In [72]:
```
df_co[(df_co['Company']=='nvnv wgzohrnvzwj otqcxwto') & (df_co['Tier_company']==3)].sort_values('CTC',ascending=False).tail(10)
```

Out[72]:

	Company	Orgyear	CTC	Position	CTC_updated_year	Email	Experience	CTC-1	CTC-2	Tier_co
23788	nvnv wgzohrnvzwj otqcxwto	2017.0	280000	Data Analyst	2021.0	c97b8f89afae9426b8540105bc855f2044a3ec4065b421...	6.0	400000.0	620000.0	
19582	nvnv wgzohrnvzwj otqcxwto	2017.0	280000	Data Analyst	2021.0	c97b8f89afae9426b8540105bc855f2044a3ec4065b421...	6.0	400000.0	620000.0	
19165	nvnv wgzohrnvzwj otqcxwto	2016.0	200000	Data Analyst	2021.0	1a19f11442abc311716e648395296a4d747426015279fb...	7.0	400000.0	620000.0	
21080	nvnv wgzohrnvzwj otqcxwto	2016.0	200000	Data Analyst	2021.0	1a19f11442abc311716e648395296a4d747426015279fb...	7.0	400000.0	620000.0	
22379	nvnv wgzohrnvzwj otqcxwto	2016.0	200000	Data Analyst	2020.0	55f4bacc1230b404e92c5edb2990e5179e031961bb14ef...	7.0	400000.0	620000.0	
20686	nvnv wgzohrnvzwj otqcxwto	2007.0	132000	Data Scientist	2017.0	3fd7b50dc84e8b2493f097c6fc33c8abed19107ed0b1d3...	16.0	400000.0	620000.0	
22878	nvnv wgzohrnvzwj otqcxwto	2015.0	100000	Data Analyst	2021.0	8815d8a3355dd3d28409080d86100195967d9a8f917dff...	8.0	400000.0	620000.0	
21160	nvnv wgzohrnvzwj otqcxwto	2020.0	7500	Data Scientist	2020.0	3175d03fd4618eb293d6f5a1d13d42a0c79f68e9acaaa3...	3.0	400000.0	620000.0	
18930	nvnv wgzohrnvzwj otqcxwto	2012.0	7000	Data Analyst	2019.0	00cd57017317c2beea73de68b4835dabd394765a045fcd...	11.0	400000.0	620000.0	
24912	nvnv wgzohrnvzwj otqcxwto	2012.0	7000	Data Analyst	2019.0	00cd57017317c2beea73de68b4835dabd394765a045fcd...	11.0	400000.0	620000.0	

Bottom 10 employees of Amazon

In [73]:
```
df_co[(df_co['Company']=='wgszxkvzn') & (df_co['Tier_company']==3)].sort_values('CTC',ascending=False).tail(10)
```

Out[73]:

	Company	Orgyear	CTC	Position	CTC_updated_year	Email	Experience	CTC-1	CTC-2	Tier_com
13159	wgszxkvzn	2019.0	300000	Data Analyst	2021.0	78cdd34bd0d29a99b3a107af275a513dad57578941e48f...	4.0	450000.0	736400.0	
13302	wgszxkvzn	2019.0	300000	Data Analyst	2021.0	b2254514508a6d38a66b7f99be7d8e7e72d3b1f1d36d4d...	4.0	450000.0	736400.0	
13318	wgszxkvzn	2019.0	300000	Data Analyst	2021.0	8c41ca512e419a10e9ef29920e00574df7a8fd8cb01a90...	4.0	450000.0	736400.0	
13556	wgszxkvzn	2016.0	200000	Data Analyst	2021.0	697697fbc6c0eb517ca877e8fd5b7564b59ab5c9b70183...	7.0	450000.0	736400.0	
12512	wgszxkvzn	2016.0	200000	Data Analyst	2021.0	697697fbc6c0eb517ca877e8fd5b7564b59ab5c9b70183...	7.0	450000.0	736400.0	
14394	wgszxkvzn	2016.0	150000	Data Analyst	2021.0	d2dfdd9d6ca1738e69e63496047baaac1e5a1972788823...	7.0	450000.0	736400.0	
14293	wgszxkvzn	2003.0	120000	Data Scientist	2018.0	0a64396c6529260a084e8b8fa8b6abdceb5aa1d46cb100...	20.0	450000.0	736400.0	
14700	wgszxkvzn	2015.0	100000	Data Analyst	2020.0	64a8668eb36d32f6c7c28fde339c0faba7323b8a814696...	8.0	450000.0	736400.0	
14110	wgszxkvzn	2007.0	94000	Data Scientist	2017.0	70b068923738c30d91d2f119f5eac1bbd9f1ee7da8609e...	16.0	450000.0	736400.0	
15082	wgszxkvzn	2014.0	4300	Data Analyst	2019.0	abf69e786daa23f50d3142b653235e2c030b3b180b07d2...	9.0	450000.0	736400.0	

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

In [74]:
```
df_comp1=df_comp[df_comp['Tier_company']==1]
df_comp1=df_comp1.sort_values(['Company','CTC'],ascending=False)
df_comp1.groupby('Company').head(10)
```

Out[74]:

	Company	Orgyear	CTC	Position	CTC_updated_year	Email	Experience	CTC-1	CTC-2	Tier_
111452	zz	2013.0	1370000	Other	2020.0	7d4588453bc463b39db8c77ef0f856957fc42f5d54cae4...	10.0	787100.0	1074200.0	
107303	zxztrtvuo	2002.0	3000000	Backend Architect	2019.0	8c7b6b29eddd85e147c7d02507735f2c250d6c40d34bb2...	21.0	551800.0	1022500.0	
107288	zxztrtvuo	2011.0	2700000	Devops Engineer	2018.0	da254908334157ddde0078c9bf967114ae67cc00816f4c...	12.0	551800.0	1022500.0	
107269	zxztrtvuo	2015.0	2500000	Frontend Engineer	2021.0	234f0f52e89b20231f5685551b865a08a5634189e1ffec...	8.0	551800.0	1022500.0	
107280	zxztrtvuo	2014.0	2500000	Frontend Engineer	2021.0	66a9ddfbd95e2f5fb57e5f649bd22eeeb7c1a9aec920b8...	9.0	551800.0	1022500.0	
...	...	...	...	...	...	...	...	...	...	
175822	123ongqto	2016.0	2560000	Backend Engineer	2020.0	160c93562f03a86198438f7b0ba964d1031e7b8c13c9d2...	7.0	1534900.0	2039800.0	
122736	10nxbto	2019.0	500000	Frontend Engineer	2019.0	abf1baa6163b89982e1a66ea5eca355109946e5030f342...	4.0	400000.0	409800.0	
122737	10nxbto	2019.0	410000	Fullstack Engineer	2020.0	2dde60f4a510b412ac4e8011e806282ecb181a51081751...	4.0	400000.0	409800.0	
178611	1 jtvq	2018.0	1700000	Backend Engineer	2019.0	034ae72dd3b77497e96173751f8805d5314bc2363e30c2...	5.0	1003200.0	1346400.0	
170712	01 ojztqsj	2011.0	830000	Frontend Engineer	2019.0	e1e15fada844f35fcc33927343d0c80f55526b87c40eee...	12.0	454800.0	639600.0	

21970 rows × 10 columns

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

In [75]:
```
df_comp1=df_comp[df_comp['Tier_company']==3]
df_comp1=df_comp1.sort_values(['Company','CTC'],ascending=False)
df_comp1.groupby('Company').tail(10)
```

Out[75]:

	Company	Orgyear	CTC	Position	CTC_updated_year	Email	Experience	CTC-1	CTC-2	Tier
152432	zzgato	2014.0	130000	Ios Engineer	2017.0	d421e52125f8057c65fa554752be03b056221c8590ff26...	9.0	130000.0	130000.0	
155789	zzb ztdnstz vacxogqj ucn rna	2017.0	600000	Fullstack Engineer	2021.0	ca8935e2314a1bac3947e60bbd2ee10524112898da29eb...	6.0	600000.0	600000.0	
155790	zzb ztdnstz vacxogqj ucn rna	2017.0	600000	Fullstack Engineer	2021.0	ca8935e2314a1bac3947e60bbd2ee10524112898da29eb...	6.0	600000.0	600000.0	
111451	zz	2009.0	500000	Backend Engineer	2021.0	d6923a6f81c7b36615d9f14349fe01aec442029b2c502f...	14.0	787100.0	1074200.0	
170716	zyvzwt wgzohrnxzs tzsxzttqo	2012.0	940000	Frontend Engineer	2019.0	49df7e15dae1895709b1c5c3a58008243f914f7ca54186...	11.0	940000.0	940000.0	
...	...	...	...	...	...	...	...	...	...	
148657	05mz exzytvrny uqxcvnt rxbxnta	2019.0	1100000	Backend Engineer	2021.0	4702229ffb6968c87b16fc57e730769e554184e322e111...	4.0	1100000.0	1100000.0	
170713	01 ojztqsj	2016.0	270000	Android Engineer	2019.0	819789ff4068fd5c8facf8a5074cdd2e1ff989c95ae02c...	7.0	454800.0	639600.0	
192088	0000	2017.0	300000	Other	2020.0	b3f3bb98cbca4b1ce5dfd5abb4e500ce6f6b66288a5202...	6.0	300000.0	300000.0	
68049	0	2020.0	100000	Other	2020.0	e80f7c9c26012bfdeca551e2b8642a93e45939d3d677c5...	3.0	100000.0	100000.0	
68050	0	2020.0	100000	Other	2020.0	e80f7c9c26012bfdeca551e2b8642a93e45939d3d677c5...	3.0	100000.0	100000.0	

55969 rows × 10 columns

Doing analysis at Company & Job Position level. Name that flag Class with values [1,2,3]

In [76]:
```
vals = {'CTC': [q1, q3]}
df6=df5.groupby(['Company','Position']).agg(vals).reset_index()
```

In [77]:
```
df6.columns=['Company','Position','CTC-1','CTC-2']
```

In [78]:
```
df_comp_pos=pd.merge(orgi, df6,  how='left', left_on=['Company','Position'], right_on = ['Company','Position'])
```

In [79]:
```
df_comp_pos['Tier_company']=0
```

In [80]:
```
for i in range(len(df_comp_pos)):
    if df_comp_pos.iloc[i]['CTC']<=df_comp_pos.iloc[i]['CTC-1']:
        df_comp_pos.iloc[i, df_comp_pos.columns.get_loc('Tier_company')]=3
    elif df_comp_pos.iloc[i]['CTC']<=df_comp_pos.iloc[i]['CTC-2']:
        df_comp_pos.iloc[i, df_comp_pos.columns.get_loc('Tier_company')]=2
    else:
        df_comp_pos.iloc[i, df_comp_pos.columns.get_loc('Tier_company')]=1
```

In [133]:
```
df_comp_pos
```

Out[133]:

	Company	Orgyear	CTC	Position	CTC_updated_year	Email	Experience	CTC-1	CTC-2	Ti
0	utqoxontzn ojontbo	2020.0	450000	Backend Engineer	2019.0	e245da546bf50eba09cb7c9976926bd56557d1ac9a17fb...	3.0	650000.0	1054800.0	
1	xrbhd	2019.0	360000	SDE Intern	2019.0	b2dc928f4c22a9860b4a427efb8ab761e1ce0015fba1a5...	4.0	360000.0	360000.0	
2	mvqwrvjo wgqugqvnt mvzpxzs	2020.0	800000	Backend Engineer	2020.0	7f24d2f5171ea469482a9966832237bc023678883ecd0c...	3.0	969000.0	1600000.0	
3	puxn	2020.0	1400000	Fullstack Engineer	2019.0	26b502eb6439ac80bd618a6f7c2b1c640b84c1e64c472c...	3.0	602900.0	817400.0	
4	mvlvl exzotqc	2018.0	100000	Other	2021.0	62d2e04b44c8bf2f6ec15d5b4c259c06199f598dc51816...	5.0	191000.0	895000.0	
...	...	...	...	...	...	...	...	...	...	...
192819	wos xzntqzvnxgzvr	2016.0	1500000	Fullstack Engineer	2021.0	ee1e251ebb54cffeddc09244bbc0ed122a1cf59511bbc1...	7.0	1224000.0	1893600.0	
192820	xzegojo	2019.0	1200000	Fullstack Engineer	2021.0	aec7061c552cfd56fe635ca0e9d347d2e63a95dbcd2ef8...	4.0	480000.0	700000.0	
192821	wgbuzgcv wgznqvwn	2015.0	1000000	Data Scientist	2021.0	251f5f33672f58c65f43e94d0fc0fef69a7f6149b3b4db...	8.0	1000000.0	1000000.0	
192822	ahzzyhbmj	2019.0	1100000	Data Scientist	2021.0	617a49afbc5efc0f1692780d20fc566ba4ed4f1125491c...	4.0	1150000.0	1298000.0	
192823	ertdnqvat ojontbo rbn	2017.0	1100000	Frontend Engineer	2021.0	8400c092bfe0b5baca38888c8cf2b9fda5e206c9812a22...	6.0	1100000.0	1100000.0	

192824 rows × 10 columns

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

In [81]:
```
df_comp_pos[(df_comp_pos['Company']=='wgszxkvzn') & (df_comp_pos['Experience'].isin([5,6,7]))
            &(df_comp_pos['Tier_company']==1)
            & (df_comp_pos['Position']=='Fullstack Engineer')].sort_values('CTC',ascending=False).head(10)
```

Out[81]:

	Company	Orgyear	CTC	Position	CTC_updated_year	Email	Experience	CTC-1	CTC-2	Tier_co
169699	wgszxkvzn	2018.0	3385000	Fullstack Engineer	2020.0	73cbc796a4302bd691bda6ad9da47ec92f366fb697869e...	5.0	500000.0	800000.0	
184818	wgszxkvzn	2018.0	2120000	Fullstack Engineer	2020.0	c78f26685af416e80267e084c1145ff0ae81b035d5020a...	5.0	500000.0	800000.0	
70651	wgszxkvzn	2018.0	1900000	Fullstack Engineer	2021.0	ee6dbaf9d76391fd7849a02ee77981ed6970c23717312e...	5.0	500000.0	800000.0	
13162	wgszxkvzn	2016.0	1800000	Fullstack Engineer	2021.0	04f6b00a81b03f9fc12a77efceba65a1241263968f2f8a...	7.0	500000.0	800000.0	
151054	wgszxkvzn	2016.0	1610000	Fullstack Engineer	2021.0	b5e6e4cf17b70efe7ff25b3653cd75cfbaae06d946afd3...	7.0	500000.0	800000.0	
28754	wgszxkvzn	2016.0	1610000	Fullstack Engineer	2021.0	b5e6e4cf17b70efe7ff25b3653cd75cfbaae06d946afd3...	7.0	500000.0	800000.0	
185959	wgszxkvzn	2018.0	1600000	Fullstack Engineer	2020.0	d89522b72bd85949b7e6e73a1d15e84ec80748f1121bfb...	5.0	500000.0	800000.0	
63870	wgszxkvzn	2018.0	1500000	Fullstack Engineer	2021.0	442a34b6957f8641685b0bc0ae4c446ae653e6b92ef4c7...	5.0	500000.0	800000.0	
120541	wgszxkvzn	2017.0	1418000	Fullstack Engineer	2021.0	af3fbd29c73843715ecb4d58c6e1a6629246ecd3ecb671...	6.0	500000.0	800000.0	
95653	wgszxkvzn	2017.0	1390000	Fullstack Engineer	2021.0	349d27bcc1b792629001b2625fef53a72c2caf7d403b59...	6.0	500000.0	800000.0	

In [82]:
```
df_comp_pos['Tier_company'].value_counts()
```

Out[82]:
```
3    104899
1     47540
2     40385
Name: Tier_company, dtype: int64
```

In [83]:
```
vals = {'CTC': [q1, q3]}
df6=df5.groupby(['Position']).agg(vals).reset_index()
```

In [84]:
```
df6.columns=['Position','CTC-1','CTC-2']
```

In [85]:
```
df6
```

Out[85]:

	Position	CTC-1	CTC-2
0	Account	2000000.0	2000000.0
1	Administrator	1409600.0	2439200.0
2	Advisory consultant uiux expert	1650000.0	1650000.0
3	Agency collection manager	360000.0	360000.0
4	Analyst consultant	23000.0	23000.0
...	...	...	...
313	Vice president	2452000.0	5056000.0
314	Voice president	2950000.0	2950000.0
315	Web engineer	1150000.0	1150000.0
316	Zomato	500000.0	500000.0
317	Zomato	100000.0	100000.0

318 rows × 3 columns

In [86]:
```
df_pos=pd.merge(orgi, df6,  how='left', left_on=['Position'], right_on = ['Position'])
```

In [87]:
```
df_pos['Tier_company']=0
for i in range(len(df_pos)):
    if df_pos.iloc[i]['CTC']<=df_pos.iloc[i]['CTC-1']:
        df_pos.iloc[i, df_pos.columns.get_loc('Tier_company')]=3
    elif df_pos.iloc[i]['CTC']<=df_pos.iloc[i]['CTC-2']:
        df_pos.iloc[i, df_pos.columns.get_loc('Tier_company')]=2
    else:
        df_pos.iloc[i, df_pos.columns.get_loc('Tier_company')]=1
```

In [88]:
```
df_pos['Tier_company'].value_counts()
```

Out[88]:
```
3    64847
2    64414
1    63563
Name: Tier_company, dtype: int64
```

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

In [139]:
```
df6=df5.groupby('Position')[['CTC','Experience']].agg({'CTC':['mean','count','min','max','median'],'Experience':['mean','count','
```

In [140]:
```
df6.columns=['Position',
             'CTC-mean',
             'CTC-count',
                    'CTC-min',
                    'CTC-max',
                    'CTC-median',
             'Experience-mean',
             'Experience-count',
             'Experience-min',
             'Experience-max',
             'Experience-median']
```

Top Positions based on CTC-count

In [141]:
```
df6.sort_values(by=['CTC-count'], axis=0, ascending=False, inplace=False, kind='quicksort', na_position='last')
```

Out[141]:

	Position	CTC-mean	CTC-count	CTC-min	CTC-max	CTC-median	Experience-mean	Experience-count	Experience-min	Experience-max	Experience-median
37	Backend Engineer	1.687718e+06	61051	2	200000000	1100000.0	7.534881	61051	1.0	28.0	7.0
96	Fullstack Engineer	1.634160e+06	35790	600	200000000	900000.0	6.813942	35790	1.0	28.0	6.0
155	Other	3.141630e+06	24395	24	200000000	600000.0	6.712687	24395	1.0	32.0	6.0
94	Frontend Engineer	1.628758e+06	13573	1000	200000000	880000.0	7.818758	13573	1.0	29.0	7.0
182	QA Engineer	1.625693e+06	7753	6	200000000	700000.0	8.860699	7753	1.0	28.0	8.0
...	...	...	...	...	...	...	...	...	...	...	...
135	Marketing professional	5.400000e+05	1	540000	540000	540000.0	16.000000	1	16.0	16.0	16.0
136	Matlab programmer	1.000000e+04	1	10000	10000	10000.0	11.000000	1	11.0	11.0	11.0
138	Member of technical staff 1	1.110000e+05	1	111000	111000	111000.0	9.000000	1	9.0	9.0	9.0
140	Member of technical staff 3	3.350000e+06	1	3350000	3350000	3350000.0	7.000000	1	7.0	7.0	7.0
317	Zomato	1.000000e+05	1	100000	100000	100000.0	22.000000	1	22.0	22.0	22.0

318 rows × 11 columns

Top Positions based on CTC-mean

In [142]:
```
df6.sort_values(by=['CTC-mean'], axis=0, ascending=False, inplace=False, kind='quicksort', na_position='last')
```

Out[142]:

	Position	CTC-mean	CTC-count	CTC-min	CTC-max	CTC-median	Experience-mean	Experience-count	Experience-min	Experience-max	Experience-median
200	Reseller	1.142857e+08	7	100000000	200000000	100000000.0	9.428571	7	8.0	11.0	9.0
44	Business man	1.000000e+08	1	100000000	100000000	100000000.0	11.000000	1	11.0	11.0	11.0
63	Data entry	1.000000e+08	1	100000000	100000000	100000000.0	10.000000	1	10.0	10.0	10.0
114	Jharkhand	1.000000e+08	1	100000000	100000000	100000000.0	2.000000	1	2.0	2.0	2.0
54	Computer scientist 2	1.000000e+08	1	100000000	100000000	100000000.0	11.000000	1	11.0	11.0	11.0
...	...	...	...	...	...	...	...	...	...	...	...
95	Frontend developer	2.700000e+04	1	27000	27000	27000.0	10.000000	1	10.0	10.0	10.0
4	Analyst consultant	2.300000e+04	1	23000	23000	23000.0	12.000000	1	12.0	12.0	12.0
148	New graduate	1.100000e+04	3	2000	16000	15000.0	8.000000	3	8.0	8.0	8.0
136	Matlab programmer	1.000000e+04	1	10000	10000	10000.0	11.000000	1	11.0	11.0	11.0
118	Junior consultant	1.000000e+04	1	10000	10000	10000.0	7.000000	1	7.0	7.0	7.0

318 rows × 11 columns

Experience and mean CTC

In [92]:
```
df6=df5.groupby('Experience')[['CTC','Experience']].agg({'CTC':['mean','count','min','max','median'],'Experience':['mean','count'
```

In [93]:
```
df6
```

Out[93]:

	Experience	CTC					Experience				
		mean	count	min	max	median	mean	count	min	max	median
0	1.0	1.065178e+07	446	10000	200000000	1000000.0	1.0	446	1.0	1.0	1.0
1	2.0	3.824527e+06	2976	1000	200000000	700000.0	2.0	2976	2.0	2.0	2.0
2	3.0	1.907053e+06	12529	24	200000000	700000.0	3.0	12529	3.0	3.0	3.0
3	4.0	1.526506e+06	21942	1000	200000000	670000.0	4.0	21942	4.0	4.0	4.0
4	5.0	1.627419e+06	23732	500	200000000	700000.0	5.0	23732	5.0	5.0	5.0
5	6.0	1.720543e+06	22115	600	200000000	750000.0	6.0	22115	6.0	6.0	6.0
6	7.0	2.029134e+06	22026	15	200000000	850000.0	7.0	22026	7.0	7.0	7.0
7	8.0	2.124149e+06	19729	1000	200000000	930000.0	8.0	19729	8.0	8.0	8.0
8	9.0	2.211956e+06	15951	2	200000000	1000000.0	9.0	15951	9.0	9.0	9.0
9	10.0	2.104089e+06	11720	6	200000000	1200000.0	10.0	11720	10.0	10.0	10.0
10	11.0	2.277118e+06	9973	600	200000000	1350000.0	11.0	9973	11.0	11.0	11.0
11	12.0	2.050241e+06	7573	1000	200000000	1500000.0	12.0	7573	12.0	12.0	12.0
12	13.0	2.310445e+06	5412	1000	200000000	1600000.0	13.0	5412	13.0	13.0	13.0
13	14.0	2.444711e+06	3507	1000	110000000	1650000.0	14.0	3507	14.0	14.0	14.0
14	15.0	2.518269e+06	2537	2000	200000000	1700000.0	15.0	2537	15.0	15.0	15.0
15	16.0	2.583633e+06	2110	1000	200000000	2000000.0	16.0	2110	16.0	16.0	16.0
16	17.0	2.592962e+06	1931	3000	101500000	2089999.0	17.0	1931	17.0	17.0	17.0
17	18.0	2.637817e+06	1733	1000	100000000	2400000.0	18.0	1733	18.0	18.0	18.0
18	19.0	3.318637e+06	1344	2000	200000000	2500000.0	19.0	1344	19.0	19.0	19.0
19	20.0	3.446484e+06	938	1500	190000000	2500000.0	20.0	938	20.0	20.0	20.0
20	21.0	3.672734e+06	629	1000	200000000	2750000.0	21.0	629	21.0	21.0	21.0
21	22.0	6.133508e+06	615	1000	100000000	2600000.0	22.0	615	22.0	22.0	22.0
22	23.0	3.920869e+06	428	3000	165000000	2700000.0	23.0	428	23.0	23.0	23.0
23	24.0	3.677797e+06	295	13000	100000000	3000000.0	24.0	295	24.0	24.0	24.0
24	25.0	2.869068e+06	219	95000	9000000	3000000.0	25.0	219	25.0	25.0	25.0
25	26.0	2.978153e+06	192	60000	16000000	2835000.0	26.0	192	26.0	26.0	26.0
26	27.0	2.848547e+06	106	21000	10500000	2775000.0	27.0	106	27.0	27.0	27.0
27	28.0	3.066754e+06	57	25000	7000000	3290000.0	28.0	57	28.0	28.0	28.0
28	29.0	3.561290e+06	31	140000	7000000	3370000.0	29.0	31	29.0	29.0	29.0
29	30.0	3.850400e+06	25	800000	7000000	3700000.0	30.0	25	30.0	30.0	30.0
30	31.0	1.500000e+08	1	150000000	150000000	150000000.0	31.0	1	31.0	31.0	31.0
31	32.0	2.000000e+08	2	200000000	200000000	200000000.0	32.0	2	32.0	32.0	32.0

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

In [94]:
```
df_new=df5.groupby(['Company','Position'])['CTC'].agg('mean').reset_index().sort_values(['Company','CTC'],ascending=False)
```

In [95]:
```
df_new.groupby('Company').head(2)
```

Out[95]:

	Company	Position	CTC
61453	zzgato	Ios Engineer	130000.0
61452	zzb ztdnstz vacxogqj ucn rna	Fullstack Engineer	600000.0
61451	zz	Other	1370000.0
61450	zz	Backend Engineer	500000.0
61449	zyvzwt wgzohrnxzs tzsxzttqo	Frontend Engineer	940000.0
...	...	...	...
4	05mz exzytvrny uqxcvnt rxbxnta	Backend Engineer	1100000.0
3	01 ojztqsj	Frontend Engineer	830000.0
2	01 ojztqsj	Android Engineer	270000.0
1	0000	Other	300000.0
0	0	Other	100000.0

45990 rows × 3 columns

In [96]:
```
df5.isnull().sum()
```

Out[96]:
```
Company             0
Orgyear             0
CTC                 0
Position            0
CTC_updated_year    0
Email               0
Experience          0
dtype: int64
```

Target encoding and scaling

In [97]:
```
df5.drop('Email',axis=1,inplace=True)
Targetenc = TargetEncoder(cols=['Company','Position'])
df = Targetenc.fit_transform(X = df5, y = df5['CTC'])
```

In [98]:
```
df
```

Out[98]:

	Company	Orgyear	CTC	Position	CTC_updated_year	Experience
0	1.023042e+06	2020.0	450000	1.687718e+06	2019.0	3.0
1	2.055971e+06	2019.0	360000	2.205992e+06	2019.0	4.0
2	1.506892e+06	2020.0	800000	1.687718e+06	2020.0	3.0
3	1.688210e+06	2020.0	1400000	1.634160e+06	2019.0	3.0
4	1.678464e+06	2018.0	100000	3.141630e+06	2021.0	5.0
...	...	...	...	...	...	...
141995	1.503929e+06	2016.0	1500000	1.634160e+06	2021.0	7.0
141996	1.405285e+06	2019.0	1200000	1.634160e+06	2021.0	4.0
141997	2.055971e+06	2015.0	1000000	1.700511e+06	2021.0	8.0
141998	1.328060e+06	2019.0	1100000	1.700511e+06	2021.0	4.0
141999	1.357100e+06	2017.0	1100000	1.628758e+06	2021.0	6.0

192824 rows × 6 columns

In [99]:
```
scaler = StandardScaler()
df=scaler.fit_transform(df)
df=pd.DataFrame(df)
df.columns=['Company','Orgyear','CTC','Position','CTC_updated_year','Experience']
```

In [100]:
```
df
```

Out[100]:

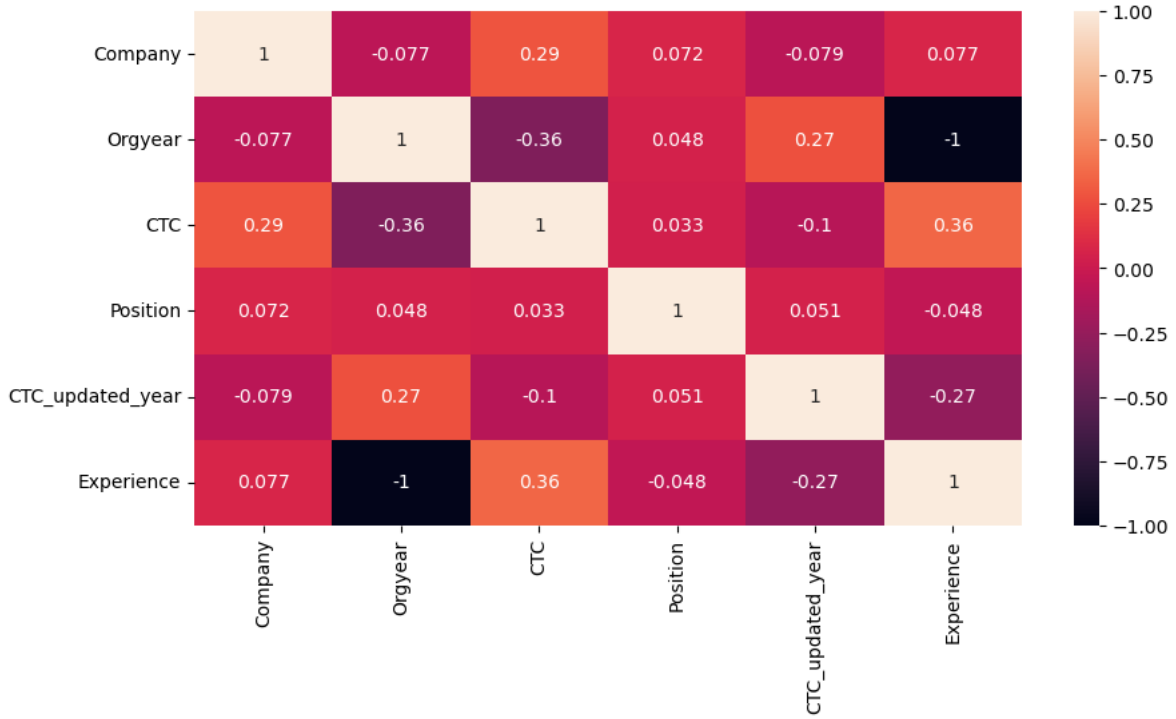
	Company	Orgyear	CTC	Position	CTC_updated_year	Experience
0	-0.666322	1.203710	-0.146097	-0.327523	-0.469712	-1.203710
1	0.094398	0.956238	-0.154284	0.136326	-0.469712	-0.956238
2	-0.309981	1.203710	-0.114257	-0.327523	0.286041	-1.203710
3	-0.176446	1.203710	-0.059674	-0.375457	-0.469712	-1.203710
4	-0.183624	0.708766	-0.177936	0.973711	1.041795	-0.708766
...	...	...	...	...	...	...
192819	-0.312164	0.213822	-0.050577	-0.375457	1.041795	-0.213822
192820	-0.384812	0.956238	-0.077868	-0.375457	1.041795	-0.956238
192821	0.094398	-0.033650	-0.096063	-0.316073	1.041795	0.033650
192822	-0.441686	0.956238	-0.086966	-0.316073	1.041795	-0.956238
192823	-0.420298	0.461294	-0.086966	-0.380292	1.041795	-0.461294

192824 rows × 6 columns

Correlation

In [146]:
```
fig,ax=plt.subplots(figsize=(10,5))
sns.heatmap(df.corr(method='spearman'),annot=True,ax=ax)
# Orgyear and experience have negative correlation
# CTC and Orgyear have negative correlation of 0.36
# CTC and Orgyear have positive correlation of 0.29
```

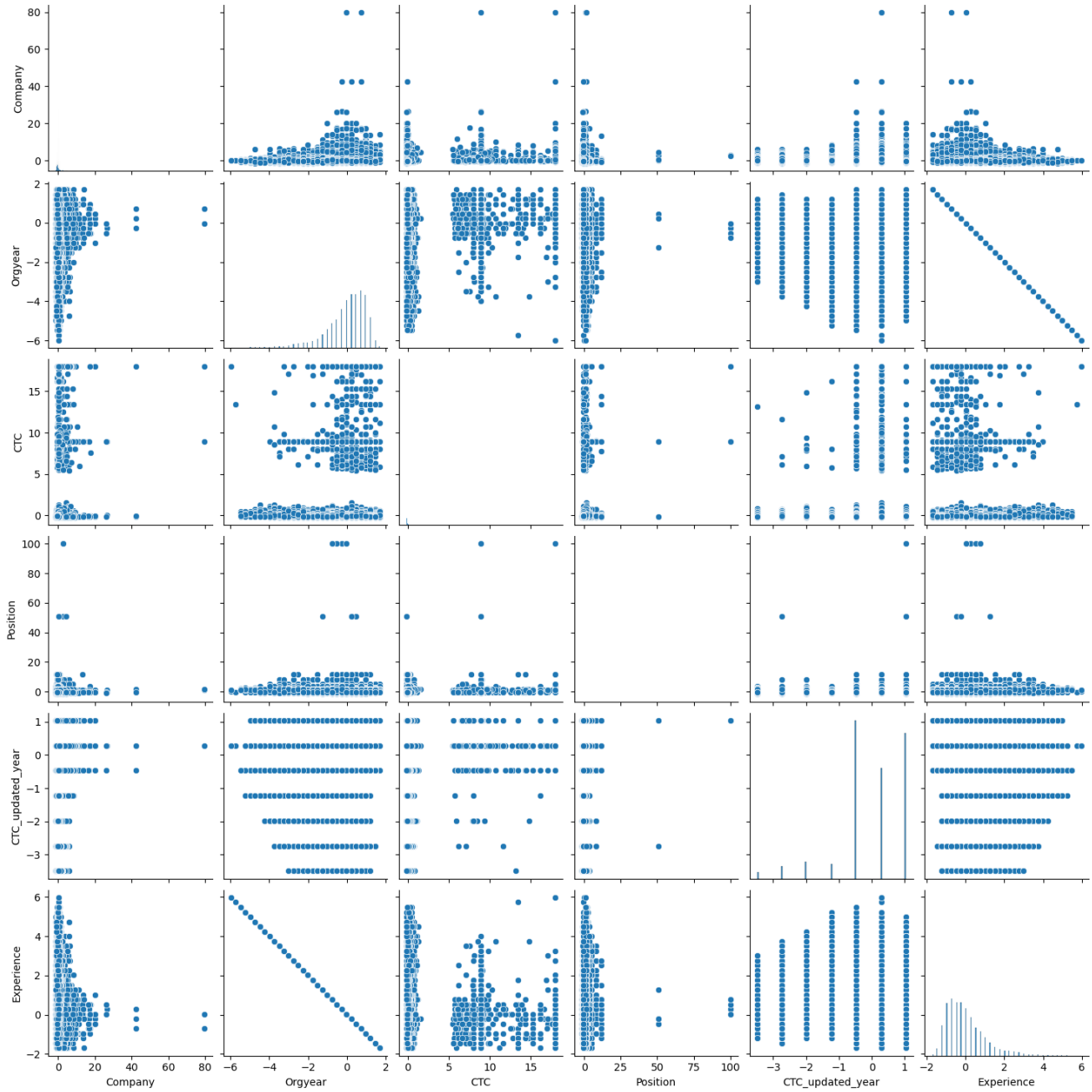
Out[146]: <AxesSubplot:>



Pairplots

In [147]:
```
sns.pairplot(df)
# With pairplots it is not clear about the number of clusters that can be formed
```

Out[147]: <seaborn.axisgrid.PairGrid at 0x1d81cb8b8e0>



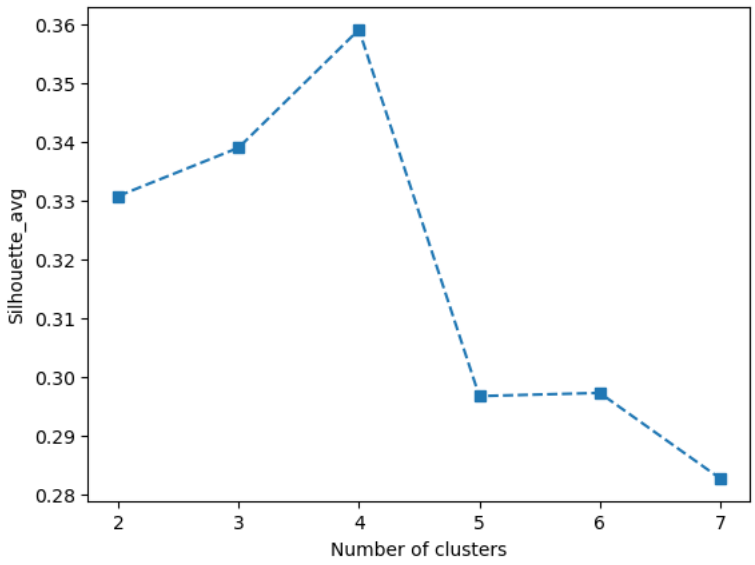
K-Means

In [101]:
```
ls=[]
si=[]
for i in range(2,8):
    kmeans=KMeans(n_clusters=i, init='k-means++', n_init=40, max_iter=500, tol=0.0001, random_state=42,algorithm='lloyd')
    kmeans.fit_transform(df)
    ls.append(kmeans.inertia_)
    y_pred=kmeans.predict(df)
    silhouette_avg = silhouette_score(df, y_pred)
    si.append(silhouette_avg)
```

In [102]:
```
ls1=[i for i in range(2,8)]
```

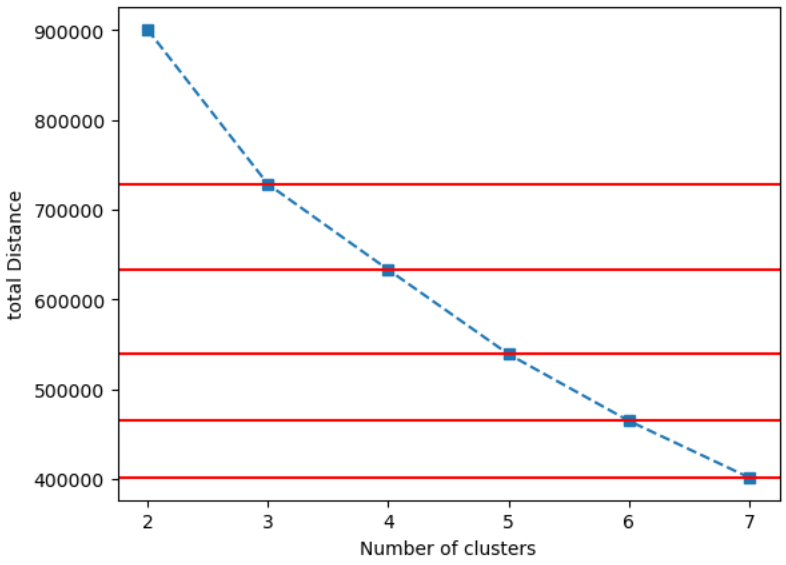
In [103]:
```
plt.plot(ls1,si,'--s')
plt.xlabel('Number of clusters')
plt.ylabel('Silhouette_avg')
# Silhouette_avg is max when number of clusters is equal to 4
```

Out[103]: Text(0, 0.5, 'Silhouette_avg')



Elbow method

In [109]:
```
plt.plot(ls1,ls,'--s')
plt.xlabel('Number of clusters')
plt.ylabel('total Distance')
for i in range(1,6):
    plt.axhline(ls[i], color = 'r', linestyle = '-')
# From below graph it is clear that drop in total distance is less when the number of cluster are >4
```



In [110]:
```
kmeans=KMeans(n_clusters=4, init='k-means++', n_init=40, max_iter=500, tol=0.0001, random_state=42,algorithm='lloyd')
kmeans.fit_transform(df)
y_pred=kmeans.predict(df)
silhouette_avg = silhouette_score(df, y_pred)
```

In [111]:
```
silhouette_avg
# silhouette_avg is 0.3591 which is greater than 0
```

Out[111]: 0.35910242133415393

In [112]:
```
pd.Series(y_pred).value_counts()
```

Out[112]:
```
0    124881
1     60989
3      5798
2      1156
dtype: int64
```

In [113]:
```
df5['Tier_kmeans']=y_pred
```

In [114]:
```
df5.groupby(['Tier_kmeans'])['CTC'].agg('mean')
```

Out[114]:
```
Tier_kmeans
0    1.031507e+06
1    1.706040e+06
2    1.339708e+08
3    1.501397e+06
Name: CTC, dtype: float64
```

In [119]:
```
di={0:4,1:2,2:1,3:3}
df5['Tier_kmeans']=df5['Tier_kmeans'].map(di)
```

Mean CTC for each tier based on kmeans

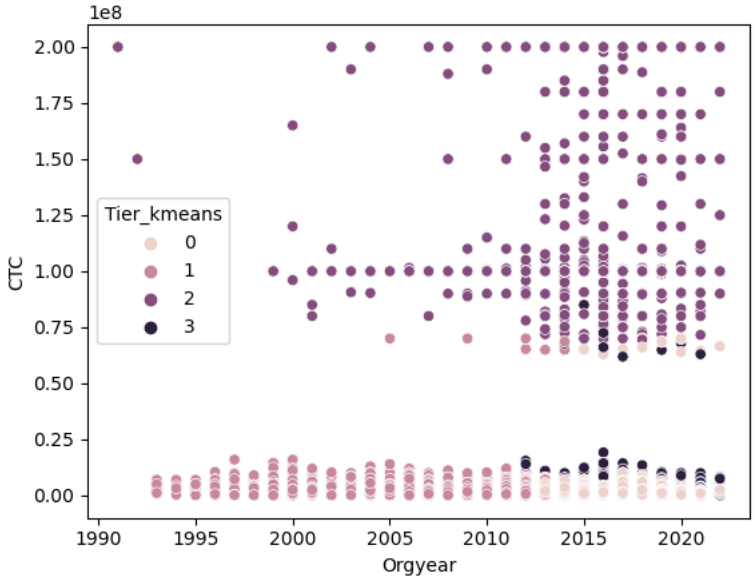
In [120]:
```
df5.groupby(['Tier_kmeans'])['CTC'].agg('mean')
```

Out[120]:
```
Tier_kmeans
1    1.339708e+08
2    1.706040e+06
3    1.501397e+06
4    1.031507e+06
Name: CTC, dtype: float64
```

Scatterplot of the clusters CTC vs Orgyear

In [115]:
```
sns.scatterplot(data=df5,x='Orgyear', y='CTC', hue='Tier_kmeans')
```

Out[115]:
```
<AxesSubplot:xlabel='Orgyear', ylabel='CTC'>
```



In [116]:
```
df
```

Out[116]:

	Company	Orgyear	CTC	Position	CTC_updated_year	Experience
0	-0.666322	1.203710	-0.146097	-0.327523	-0.469712	-1.203710
1	0.094398	0.956238	-0.154284	0.136326	-0.469712	-0.956238
2	-0.309981	1.203710	-0.114257	-0.327523	0.286041	-1.203710
3	-0.176446	1.203710	-0.059674	-0.375457	-0.469712	-1.203710
4	-0.183624	0.708766	-0.177936	0.973711	1.041795	-0.708766
...	...	...	...	...	...	...
192819	-0.312164	0.213822	-0.050577	-0.375457	1.041795	-0.213822
192820	-0.384812	0.956238	-0.077868	-0.375457	1.041795	-0.956238
192821	0.094398	-0.033650	-0.096063	-0.316073	1.041795	0.033650
192822	-0.441686	0.956238	-0.086966	-0.316073	1.041795	-0.956238
192823	-0.420298	0.461294	-0.086966	-0.380292	1.041795	-0.461294

192824 rows × 6 columns

Hierarchical clustering

In [152]:
```
df1=df.sample(n=int(len(df)*0.3))
```

In [153]:
```
df1
```

Out[153]:

	Company	Orgyear	CTC	Position	CTC_updated_year	Experience
163511	-1.100657	0.708766	-0.141548	-0.375457	1.041795	-0.708766
160102	-0.111366	0.213822	-0.032383	-0.327523	0.286041	-0.213822
105204	0.804959	-0.033650	0.022199	-0.327523	-0.469712	0.033650
121100	0.470881	-1.271009	0.067685	-0.316073	-0.469712	1.271009
112651	-0.639061	-0.281122	-0.059674	-0.327523	1.041795	0.281122
...	...	...	...	...	...	...
3683	-0.084124	-0.528593	-0.050577	-0.327523	0.286041	0.528593
164341	-0.252451	0.461294	-0.169931	0.973711	-0.469712	-0.461294
132376	0.094398	-1.271009	-0.114257	-0.380292	-0.469712	1.271009
190397	0.842509	0.708766	0.176850	-0.327523	-0.469712	-0.708766
44565	0.985081	-2.260896	0.631704	1.319978	-1.981219	2.260896

57847 rows × 6 columns

In [154]:
```
cl = AgglomerativeClustering(n_clusters=3, linkage='ward').fit(df1)
```

In [155]:
```
pd.Series(cl.labels_).value_counts()
```

Out[155]:
```
1    41812
0    15709
2      326
dtype: int64
```

In [156]:
```
df6=df5.iloc[df1.index]
```

In [157]:
```
df6['Tier_hcl']=cl.labels_
```

In [158]:
```
df6
```

Out[158]:

	Company	Orgyear	CTC	Position	CTC_updated_year	Experience	Tier_kmeans	Tier_hcl
112687	rgsxwrvaatq ntwyzgrgsxto	2018.0	500000	Fullstack Engineer	2021.0	5.0	4	1
109278	btaxvztn	2016.0	1700000	Backend Engineer	2020.0	7.0	4	1
54380	erxupvqn	2015.0	2300000	Backend Engineer	2019.0	8.0	4	1
70276	uvjuvr	2010.0	2800000	Data Scientist	2019.0	13.0	2	0
61827	gqvzst mhoxztoo otqcxwto	2014.0	1400000	Backend Engineer	2021.0	9.0	4	1
...	...	...	...	...	...	...	...	...
3683	uvjnb	2013.0	1500000	Backend Engineer	2020.0	10.0	2	0
113517	ywr ntwyzgrgsxto	2017.0	188000	Other	2019.0	6.0	4	1
81552	xn wgatqo	2010.0	800000	Frontend Engineer	2019.0	13.0	2	0
139573	bxwqgogen	2018.0	4000000	Backend Engineer	2019.0	5.0	4	1
44565	vbvkgz	2006.0	9000000	Engineering Leadership	2017.0	17.0	2	0

57847 rows × 8 columns

In [159]:
```
df6.groupby(['Tier_hcl'])['CTC'].agg('mean')
```

Out[159]:
```
Tier_hcl
0    1.685100e+06
1    1.080494e+06
2    1.327912e+08
Name: CTC, dtype: float64
```

In [160]:
```
di={0:2,1:3,2:1}
```

In [161]:
```
df6['Tier_hcl']=df6['Tier_hcl'].map(di)
```

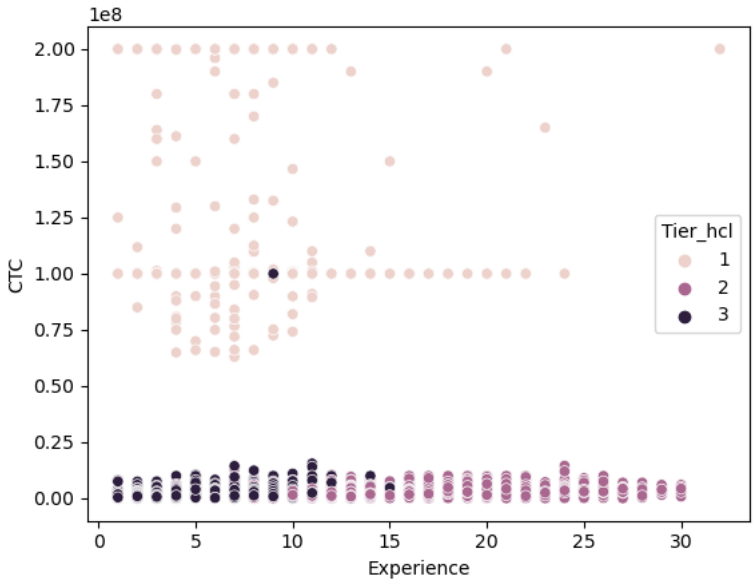
In [162]:
```
df6.groupby(['Tier_hcl'])['CTC'].agg('mean')
# Mean of CTC based on each tier
```

Out[162]:
```
Tier_hcl
1    1.327912e+08
2    1.685100e+06
3    1.080494e+06
Name: CTC, dtype: float64
```

Scatterplot of the clusters CTC vs Experience

In [164]:
```
sns.scatterplot(data=df6,x='Experience', y='CTC', hue='Tier_hcl')
```

Out[164]:
```
<AxesSubplot:xlabel='Experience', ylabel='CTC'>
```



In [163]:
```
df6
```

Out[163]:

	Company	Orgyear	CTC	Position	CTC_updated_year	Experience	Tier_kmeans	Tier_hcl
112687	rgsxwrvaatq ntwyzgrgsxto	2018.0	500000	Fullstack Engineer	2021.0	5.0	4	3
109278	btaxvztn	2016.0	1700000	Backend Engineer	2020.0	7.0	4	3
54380	erxupvqn	2015.0	2300000	Backend Engineer	2019.0	8.0	4	3
70276	uvjuvr	2010.0	2800000	Data Scientist	2019.0	13.0	2	2
61827	gqvzst mhoxztoo otqcxwto	2014.0	1400000	Backend Engineer	2021.0	9.0	4	3
...	...	...	...	...	...	...	...	...
3683	uvjnb	2013.0	1500000	Backend Engineer	2020.0	10.0	2	2
113517	ywr ntwyzgrgsxto	2017.0	188000	Other	2019.0	6.0	4	3
81552	xn wgatqo	2010.0	800000	Frontend Engineer	2019.0	13.0	2	2
139573	bxwqgogen	2018.0	4000000	Backend Engineer	2019.0	5.0	4	3
44565	vbvkgz	2006.0	9000000	Engineering Leadership	2017.0	17.0	2	2

57847 rows × 8 columns