SCALER - Clustering in Learner Profiling

Scaler is an online tech-versity offering intensive computer science & Data Science courses through live classes delivered by tech leaders and subject matter experts. 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.

Following are the columns in the dataset.

- 'Unnamed 0' - Index of the dataset
- Email_hash - Anonymised Personal Identifiable Information (PII)
- Company_hash - This represents an anonymized identifier for the company, which is the current employer of the learner.
- orgyear - Employment start date
- CTC - Current CTC
- Job_position - Job profile in the company
- CTC_updated_year - Year in which CTC got updated (Yearly increments, Promotions)

Aim is to leverage data science and unsupervised learning, particularly clustering techniques so that Scaler can group learners with similar profiles, especially in terms of their current roles, companies, and experience aiding in delivering a more personalized learning journey.

1. Basic data cleaning and exploration:

a.Importing data and finding the shape.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

df=pd.read_csv(r"C:\Users\devip\Desktop\Scaler projects\Scaler clustering\scaler_clustering.csv")
df.head()
```

Out[1]:

	Unnamed: 0	company_hash	email_hash	orgyear	ctc	job_position	ctc_updated_yea
0	0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	1100000	Other	2020
1	1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018.0	449999	FullStack Engineer	2019
2	2	ojzwnvwnxw vx	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2015.0	2000000	Backend Engineer	2020
3	3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	2017.0	700000	Backend Engineer	2019
4	4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	2017.0	1400000	FullStack Engineer	2019

In [2]:
```
df.columns
```

Out[2]:
```
Index(['Unnamed: 0', 'company_hash', 'email_hash', 'orgyear', 'ctc',
       'job_position', 'ctc_updated_year'],
      dtype='object')
```

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

Out[3]:
```
(205843, 7)
```

In [4]:
```
df.info()
```

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

Dropping column Unnamed as it affects the model.

In [5]:
```
df=df.drop('Unnamed: 0',axis=1)
```

Converting columns orgyear and ctc_updated_year to datetime.

In [6]:
```
# df['orgyear'] = df['orgyear'].astype(int)
# df['ctc_updated_year'] = df['ctc_updated_year'].astype(int)

# df['orgyear'] = pd.to_datetime(df['orgyear'],format='%Y')
# df['ctc_updated_year'] = pd.to_datetime(df['ctc_updated_year'],format='%Y')
```

b. Checking for null values.

In [7]:
```
df.isna().sum(axis=0)
```

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(axis=0)/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
```

Columns 'job_position' is having 25% missing values. 'orgyear' and 'company_hash' are also having null values.

c. Describing the dataset

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','category'])
```

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

d. Checking for duplicate rows.

In [11]:
```
df[df.duplicated()]
```

Out[11]:

	company_hash	email_hash	orgyear	ctc	job_position	ctc_updated_year
97138	wtqtzwt xzw	bb8e4b09544daf1bfc8c7bb9a9ae1fee35490cf3f321b8...	2014.0	1000000	FullStack Engineer	2019.0
98085	2020	6ad3e6ab27462c2c7428fa5d51405933335341d4d969b5...	2020.0	720000	NaN	2019.0
102600	voxvz uvxzno	c7fac937a34f7ae432ff1d77466eb7ea6cf25dfd5ebcca...	2020.0	1280000	NaN	2019.0
109324	wgbwvon mhoxztoo	0442a51ef5080d7d40721c007131a1d5bdeabae2c1b153...	2016.0	700000	NaN	2019.0
111354	uyxrxuo xzzgcvnxgz wvbuho	704d69965035d1c341b06fc5d83bf1b714f1625c0cf271...	2017.0	850000	iOS Engineer	2019.0
111521	aqggb ntwyzgrgsj	df81dac132d66a42a0c71a4799e1040731738e542c81ff...	2017.0	1270000	FullStack Engineer	2019.0
115241	rgfto wgbuvzxto xzw	ea363e930dabe0fbb63438e07775af3cb3b32639947c47...	2017.0	1100000	Backend Engineer	2019.0
117246	xatbxv	f451ceee50b1bfa3dc749c6aa8634ab3851a4ab961b003...	2019.0	640000	NaN	2019.0
117549	exzdtqv	e7df851527dd6f8ec95d5e13d9fb2a7255380245b808e3...	2020.0	1500000	NaN	2020.0
120371	avnvbtnxwv ogrhnxgzo uqxcvnt rxbxnta	15d7dd6801fb7cb980e77c420dd9bef5773e7ef57f510c...	2016.0	1300000	Backend Engineer	2020.0
121946	oguqv ontqxv	f48d4cd35091adb89c8e82b8bc39b68416e2e954e406fd...	2016.0	1250000	Data Scientist	2019.0
122316	eqtoytq	567e7ff3ad74ce235a75b1feea224204d35cd698922e59...	2018.0	900000	Backend Engineer	2019.0
130495	xatbxv	80a04f3eb89aa385e32b6e1c9a0b564730274632fad4c4...	2017.0	409999	Backend Engineer	2020.0
138371	xicxv	d0e72d551c69a2f9d96914515aeef797f4989b54c90ef0...	2014.0	1200000	FullStack Engineer	2019.0
141686	uhmrxwxo ovuxtzn	f27a6a759a02e90ebd17041fb26b72d13420d53edcdc99...	2020.0	940000	NaN	2019.0
143061	vwwtznhqt ogrhnxgzo uqxcvnt rxbxnta	bf09ce2b61e3bba0846412cf76b2e408c92384b373f709...	2014.0	800000	Android Engineer	2019.0
146097	axvouvqp xzw	8e5fe3154be66d7cd8730224318d913ecd10ec5197e20a...	2017.0	1000000	Backend Engineer	2021.0
151473	rgfto wgbuvzxto xzw	f67d3be9653bca997a75c81a88e851bcf0368fd83255aa...	2017.0	1265000	Backend Engineer	2019.0
157950	ti ntwyzgrgsxw	843a5216e56e06b9d31d35e0c3820beec3af19dc4978af...	2019.0	850000	FullStack Engineer	2020.0
161251	avnvftvct ucn rna	5083a995fa1623fd7d329766f8e7adbe5497a8c3c826f9...	2018.0	800000	Backend Engineer	2019.0
164554	ng nyt ztf	7b47ee99ce695d48d18dea36d3c6cc73e3b5b40ed477cf...	2019.0	450000	NaN	2020.0
165326	uhmrxwxo ovuxtzn	d40b483baf912b9f21cd1952e8b79388ce88ed5222d3d8...	2019.0	1200000	NaN	2019.0
171421	fyvnexd	7e2ac7c6b9051177ea51af3f7c8e934d6d3ce15a5cb587...	2020.0	1300000	FullStack Engineer	2020.0
175942	tdnqvbvqpo	82b93606127fa5ed0d28cb32469d7ba177b8e70088608c...	2019.0	350000	NaN	2020.0
179858	buyvoxo rna	bd443574985b2f72a4a382b6be392db2358158761f38de...	2016.0	750000	FullStack Engineer	2020.0
180630	uhmrxwxo ovuxtzn	59e67f9f149ede96889afacb1a70645fd3f309e3a1fa43...	2019.0	1620000	NaN	2019.0
182531	xznqvrxzp	c2c34a82a91169e2523727f7f15a4cc64f973ccb895b69...	2016.0	6730000	Backend Engineer	2019.0
195375	souvzz ntwyzgrgsxto xzw	31fefa78a0f32b56c8f0d60d2355d92c480b4ba95fcd83...	2018.0	600000	Support Engineer	2020.0
196492	2020	b6a63b76c3a1a395f7c3d509f2760d83aeb6e8c53db2b1...	2020.0	2700000	NaN	2019.0
196971	2020	77a5cecd2ed9bb764df8bf6da78a0ae2aef97fc87e913e...	2020.0	1000000	NaN	2019.0
201165	xzzgcvwwtq	5d00f5560a82d5ed91708273f9190499a6405abff35ab1...	2020.0	1300000	NaN	2019.0
203257	uhmrxwxo ovuxtzn	9efbaf1f3740b6661adb699ed5ee03ba10c51f6185e681...	2015.0	1500000	NaN	2019.0
205733	uhmrxwxo ovuxtzn	da614aea4d5dfacac3a2a6523e7e94b485fa3ba803db79...	2020.0	990000	NaN	2019.0

Removing the duplicate columns by keeping the first occurance.

In [12]:
```
df.drop_duplicates(keep='first', inplace=True)
```

In [13]:
```
df[df.duplicated()]
```

Out[13]:

company_hash	email_hash	orgyear	ctc	job_position	ctc_updated_year

---

2. Aggregating the dataset on email_hash .

We aggregate data on column email_hash in order to remove multiple occurrences of same learner.

In [14]:
```
df['email_hash'].value_counts().sort_values(ascending=False)
```

Out[14]:
```
bbace3cc586400bbc65765bc6a16b77d8913836cfc98b77c05488f02f5714a4b    10
6842660273f70e9aa239026ba33bfe82275d6ab0d20124021b952b5bc3d07e6c     9
298528ce3160cc761e4dc37a07337ee2e0589df251d73645aae209b010210eee     9
3e5e49daa5527a6d5a33599b238bf9bf31e85b9efa9a94f1c88c5e15a6f31378     9
b4d5afa09bec8689017d8b29701b80d664ca37b83cb883376b2e95191320da66     8
                                                                    ..
9dacdf828a4eed8fb8c78b653442621870241557933364234e2d52eb88010e5fc     1
81ef3c26c45912a491ec164c79e6bef426af9b9acc6175a7dc647ee3bd4ca0bf     1
c54186b3ff22353234e42a65e9bdf9435be3edf493ec207eb446865f3184b97b     1
f8ac4ef80618f6c618941689ce3c67e06edfb93d5423abfc49df5c6497e54968     1
0bcfc1d05f2e8dc4147743a1313aa70a119b41b30d4a1f7e738a6a87d3712c31     1
Name: email_hash, Length: 153443, dtype: int64
```

In [15]:
```
from scipy.stats import mode



def mode_data(series):
    filtered_series = series.dropna()
    if not filtered_series.empty:
        return filtered_series.mode()[0]
    return np.nan

if __name__ == "__main__":

    df_agg = df.groupby('email_hash').agg({
        'ctc': 'max',
        'company_hash': mode_data,
        'job_position': mode_data,
        'orgyear': 'min',
        'ctc_updated_year': 'max'
    })
```

In [16]:
```
df_agg=df_agg.reset_index()
df_agg
```

Out[16]:

	email_hash	ctc	company_hash	job_position	orgyear	ctc_updated_year
0	00003288036a44374976948c327f246fdbdf0778546904...	3500000	bxwqgogen	Backend Engineer	2012.0	2019.0
1	0000aaa0e6b61f7636af1954b43d294484cd151c9b3cf6...	250000	nqsn axsxnvr	Backend Engineer	2013.0	2020.0
2	0000d58fbc18012bf6fa2605a7b0357d126ee69bc41032...	1300000	gunhb	FullStack Engineer	2021.0	2019.0
3	000120d0c8aa304fcf12ab4b85e21feb80a342cfea03d4...	2000000	bxwqgotbx wgqugqvnxgz	FullStack Engineer	2004.0	2021.0
4	00014d71a389170e668ba96ae8e1f9d991591acc899025...	3400000	fvrbvqn rvmo	NaN	2009.0	2018.0
...	...	...	...	...	...	...
153438	fffc254e627e4bd1bc0ed7f01f9aebbba7c3cc56ac914e...	3529999	tqxwoogz ogenfvqt wvbuho	QA Engineer	2004.0	2019.0
153439	fffcf97db1e9c13898f4eb4cd1c2fe862358480e104535...	1600000	trnqvcg	NaN	2015.0	2018.0
153440	fffe7552892f8ca5fb8647d49ca805b72ea0e9538b6b01...	900000	znn avnv srgmvr atrxctqj otqcxwto	Devops Engineer	2014.0	2019.0
153441	ffff49f963e4493d8bbc7cc15365423d84a767259f7200...	700000	zwq wgqugqvnxgz	FullStack Engineer	2020.0	2020.0
153442	ffffa3eb3575f43b86d986911463dce7bcadcea227e5a4...	1500000	sgrabvz ovwyo	FullStack Engineer	2018.0	2021.0

153443 rows × 6 columns

In [17]:
```
df_agg.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 153443 entries, 0 to 153442
Data columns (total 6 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   email_hash        153443 non-null  object 
 1   ctc               153443 non-null  int64  
 2   company_hash      153411 non-null  object 
 3   job_position      133219 non-null  object 
 4   orgyear           153365 non-null  float64
 5   ctc_updated_year  153443 non-null  float64
dtypes: float64(2), int64(1), object(3)
memory usage: 7.0+ MB
```

In [18]:
```
(df_agg.isna().sum(axis=0)/len(df_agg))*100
```

Out[18]:
```
email_hash           0.000000
ctc                  0.000000
company_hash         0.020855
job_position        13.180139
orgyear              0.050833
ctc_updated_year     0.000000
dtype: float64
```

---

3. Feature Engineering steps.

a. Creating column Experience .

A column 'Experience' is created which is the difference between current year and orgyear.

As the column orgyear is having very extreme values, so dropping rows having extreme values.

```
In [19]: df_agg[(df_agg['orgyear'] < 1980) | (df_agg['orgyear'] > 2024)]
```

Out[19]:

	email_hash	ctc	company_hash	job_position	orgyear	ctc_updated_year
3018	050a5f7e04009ad2554fe374e4512d9dbfd30450410666...	150000	lvj vbmt	Devops Engineer	2025.0	2021.0
3991	069308440811d578c817c05392f97e8919baac6aa12aa3...	2900000	vaxnjv mxqrv wvuxnvr	Data Scientist	1.0	2019.0
6402	0a5e691a0f8c2c06862ef19d43dc11c22f462f800db26b...	800000	vxqvoxv	NaN	0.0	2019.0
7927	0ceab34736c0ba43f541a9d62f5f8ffe33f4c306ea73a5...	270000	otwhqt mrxzp	SDET	2026.0	2021.0
15444	1978da71c14333352d051bfb6054904770b70cecce389d...	400000	vzshrvq atcqrgutq	Devops Engineer	91.0	2021.0
...	...	...	...	...	...	...
138035	e66b927f4ee3bd0d7202bbd35486d23d68555fc03dcd54...	140000	hzxctqoxnj ge zgqny ntdvo	Engineering Leadership	1970.0	2020.0
138492	e725ad631cdc4c57a354f59c98b6441f0672c6b7bb8adb...	730000	bvzyvnnvz voogwxvnto	Backend Engineer	83.0	2019.0
144306	f0c712df5b5e6698a7558311dff87d2b2b4aaa12839915...	100000000	otre tburgjta	Other	2029.0	2021.0
147725	f648fa217922f5a36b510df6346a2041a3483e21289069...	1200000	mrvwpmhwp	NaN	2101.0	2021.0
152821	fee9df1faa9d4a38bb97185bb9af6687cba48b514f5d04...	880000	vbagwo	Backend Engineer	2026.0	2021.0

80 rows × 6 columns

```
In [20]: df_agg=df_agg.loc[(df_agg['orgyear'] >= 1980) & (df_agg['orgyear'] <= 2024)].copy()

         df_agg.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 153285 entries, 0 to 153442
Data columns (total 6 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   email_hash        153285 non-null  object 
 1   ctc               153285 non-null  int64  
 2   company_hash      153253 non-null  object 
 3   job_position      133102 non-null  object 
 4   orgyear           153285 non-null  float64
 5   ctc_updated_year  153285 non-null  float64
dtypes: float64(2), int64(1), object(3)
memory usage: 8.2+ MB
```

```
In [21]: df_agg['orgyear'].describe()
```

```
Out[21]: count    153285.000000
         mean       2014.808109
         std           4.357146
         min        1981.000000
         25%        2013.000000
         50%        2016.000000
         75%        2018.000000
         max        2024.000000
         Name: orgyear, dtype: float64
```

```
In [22]: df_agg.loc[:,'Experience']=2024-df_agg['orgyear']
```

```
In [23]: df_agg['Experience'].describe()
```

```
Out[23]: count    153285.000000
         mean          9.191891
         std           4.357146
         min           0.000000
         25%           6.000000
         50%           8.000000
         75%          11.000000
         max          43.000000
         Name: Experience, dtype: float64
```

b. Creating column Income bins .

In [24]:
```
df_agg['job_position']=df_agg['job_position'].str.lower()
df_agg['Income_bin']=df_agg['ctc'].apply(lambda x: 'Low' if x<1000000 else ('Medium' if x<3000000 else 'High'))
```

If CTC is below 1000000 it is marked as category 'Low', if income is below 3000000 its 'Medium' and employees having income greater than 3000000 is included in income bin 'High'.

c. Creating column Job_position_prominance .

If mean CTC of the job_position is below 1500000 it is marked as category 'Less_prominant', if income is below 4000000 its 'Medium_prominant' and job positions having income greater than 4000000 is included in bin 'Highly prominant'.

In [25]:
```
job_ctc_mean = df_agg.groupby('job_position').ctc.mean()

df_agg['Job_prominance'] = df_agg['job_position'].map(job_ctc_mean)

df_agg['Job_prominance'] = df_agg['Job_prominance'].apply(lambda x: 'Less_prominant' if pd.notna(x) and x < 150(
              ('Medium_prominant' if pd.notna(x) and x < 4000000 else  ('Highly_prominant' if pd.notna(x) else ı

df_agg.head()
```

Out[25]:

	email_hash	ctc	company_hash	job_position	orgyear	ctc_updated_year	Experien
0	00003288036a44374976948c327f246fdbdf0778546904...	3500000	bxwqgogen	backend engineer	2012.0	2019.0	1:
1	0000aaa0e6b61f7636af1954b43d294484cd151c9b3cf6...	250000	nqsn axsxnvr	backend engineer	2013.0	2020.0	1
2	0000d58fbc18012bf6fa2605a7b0357d126ee69bc41032...	1300000	gunhb	fullstack engineer	2021.0	2019.0	:
3	000120d0c8aa304fcf12ab4b85e21feb80a342cfea03d4...	2000000	bxwqgotbx wgqugqvnxgz	fullstack engineer	2004.0	2021.0	2(
4	00014d71a389170e668ba96ae8e1f9d991591acc899025...	3400000	fvrbvqn rvmo	NaN	2009.0	2018.0	1:

In [26]:
```
df_agg[df_agg['Job_prominance']=='Less_prominant']
```

Out[26]:

	email_hash	ctc	company_hash	job_position	orgyear	ctc_updated_year	Ex
8	00037a2e4fcfe2830d91270102aaaf105a324a3ce17075...	1800000	ko	sdet	2012.0	2021.0	
20	000abcc4ba53bffb10a940bbb1a02dbd641ac9248849ac...	1400000	yxpt btootzstq	sdet	2015.0	2018.0	
41	001061f980d1ac33489c1f85b1587af347bf0203ee5321...	720000	ftrro evqsg	sdet	2015.0	2019.0	
105	002c8de23775649daec5935d73d82100ae46b594c2531a...	620000	exzvonqv mvzsvrgqt	sdet	2014.0	2019.0	
109	002f81f3350685a057d429b173fca3589384be9338e163...	320000	wvustbxzx	sdet	2016.0	2017.0	
...	...	...	...	...	...	...	
153131	ff6d5467e9abf203252ac540e360fe87239efb3a3d47d5...	640000	qtertdxo ojontbo xzw	sdet	2016.0	2019.0	
153139	ff70bb2130aeb865572574048d01d14b5da1fde50c5c67...	2000000	vagmt	sdet	2011.0	2020.0	
153286	ffb57767385843c9b24b1122d7daf807afab1add9df31f...	610000	ftrro evqsg wtznqt	sdet	2014.0	2019.0	
153392	ffe7dca601ec396d1dba95854a7b8554539eab53a77751...	300000	w tast ntwyzgrgsj ucn rna	sdet	2014.0	2017.0	
153433	fffa648871d5cd698ed19605344181ad80bb19d3cf4e99...	450000	vaatwg	sdet	2014.0	2019.0	

5162 rows × 9 columns

d. Creating column current_year_ctc_updated .

Flag is 1 for employees having ctc updated in the latest year in dataset which is 2021 else 0.

In [27]:

```
df_agg['current_year_ctc_updated']=df_agg['ctc_updated_year'].apply(lambda x: 1 if x==2021 else 0)
```

---

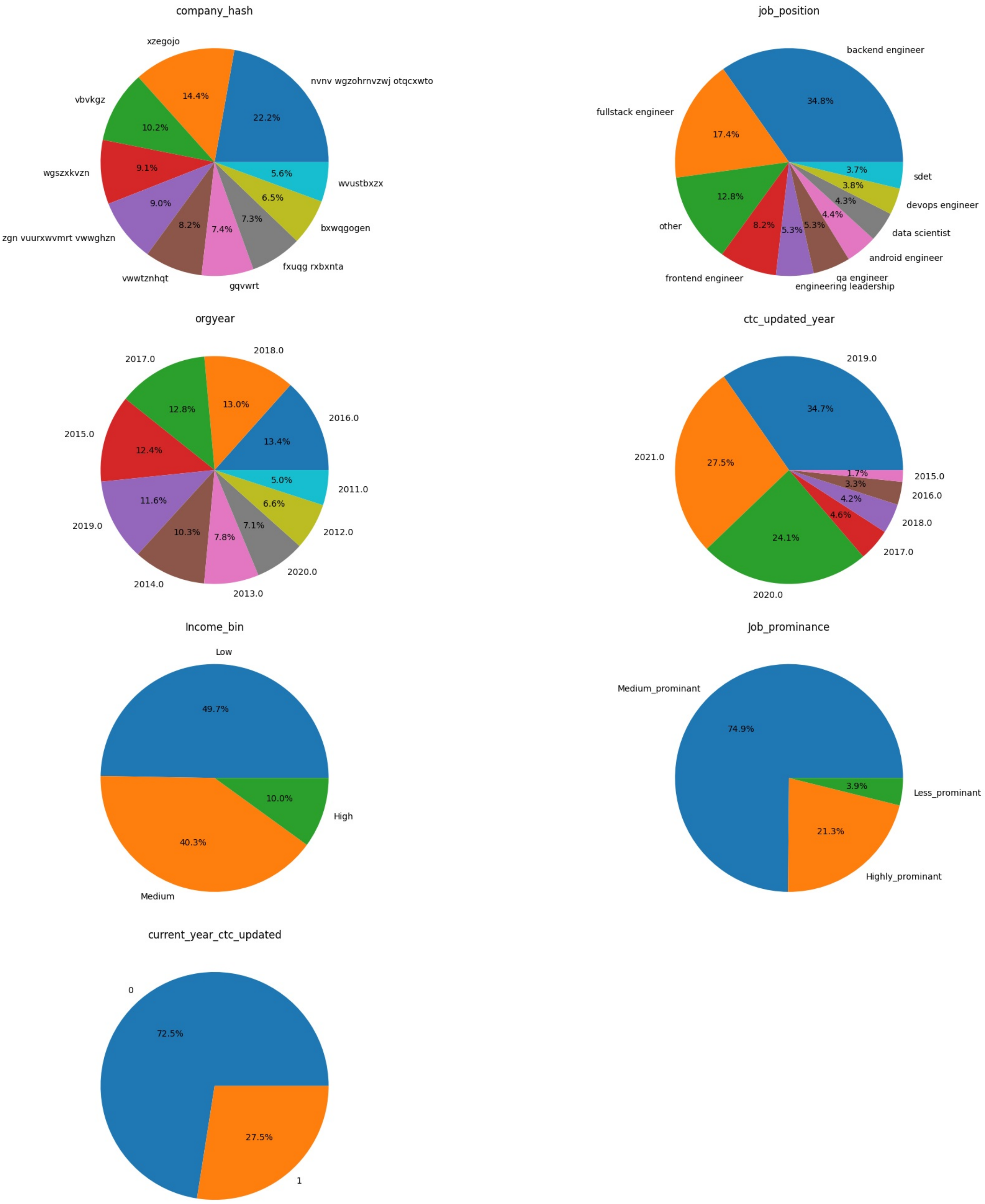
4. Use Non-graphical and graphical analysis for getting insights about variables.

a. Categorical columns analysis.

In [28]:

```
from IPython.display import display
no=1
cat_cols=['company_hash', 'job_position', 'orgyear', 'ctc_updated_year','Income_bin','Job_prominance','current_
#,'Income_bin','Job_prominance','current_year_ctc_updated'
```

```
plt.figure(figsize=(20,20))
for i in cat_cols:
    plt.subplot(4,2,no)

    plt.title(i)
    datacol=df_agg.groupby(i).size().sort_values(ascending=False).head(10)


    plt.pie(x=datacol,labels=datacol.index,autopct='%1.1f%%')

    #display(datacol.to_frame(name='count').reset_index())
    #print('\n\n')
    no+=1
plt.tight_layout()
plt.show()
```



In [29]:
```
for col in cat_cols:

    groupcol=df_agg.groupby(col).size().sort_values(ascending=False).head(10)


    plt.figure(figsize=(8,6))
    #plt.title(i)
    #plt.subplot(4,1,n)
    sns.barplot(x=groupcol.index,y=groupcol.values, order=groupcol.index)
    for index,value in enumerate(groupcol):
        plt.text(index,value,value)
    display(groupcol.to_frame(name='count').reset_index())
    plt.xticks(rotation=90)
plt.tight_layout()
plt.show()



plt.show()
```

	company_hash	count
0	nvnv wgzohrnvzwj otqcxwto	5330
1	xzegojo	3458
2	vbvkgz	2458
3	wgszxkvzn	2183
4	zgn vuurxwvmrt vwwghzn	2163
5	vwwtznhqt	1964
6	gqvwrt	1766
7	fxuqg rxbxnta	1753
8	bxwqgogen	1570
9	wvustbxzx	1348

	job_position	count
0	backend engineer	40068
1	fullstack engineer	20039
2	other	14752
3	frontend engineer	9389
4	engineering leadership	6133
5	qa engineer	6104
6	android engineer	5111
7	data scientist	4914
8	devops engineer	4328
9	sdet	4317

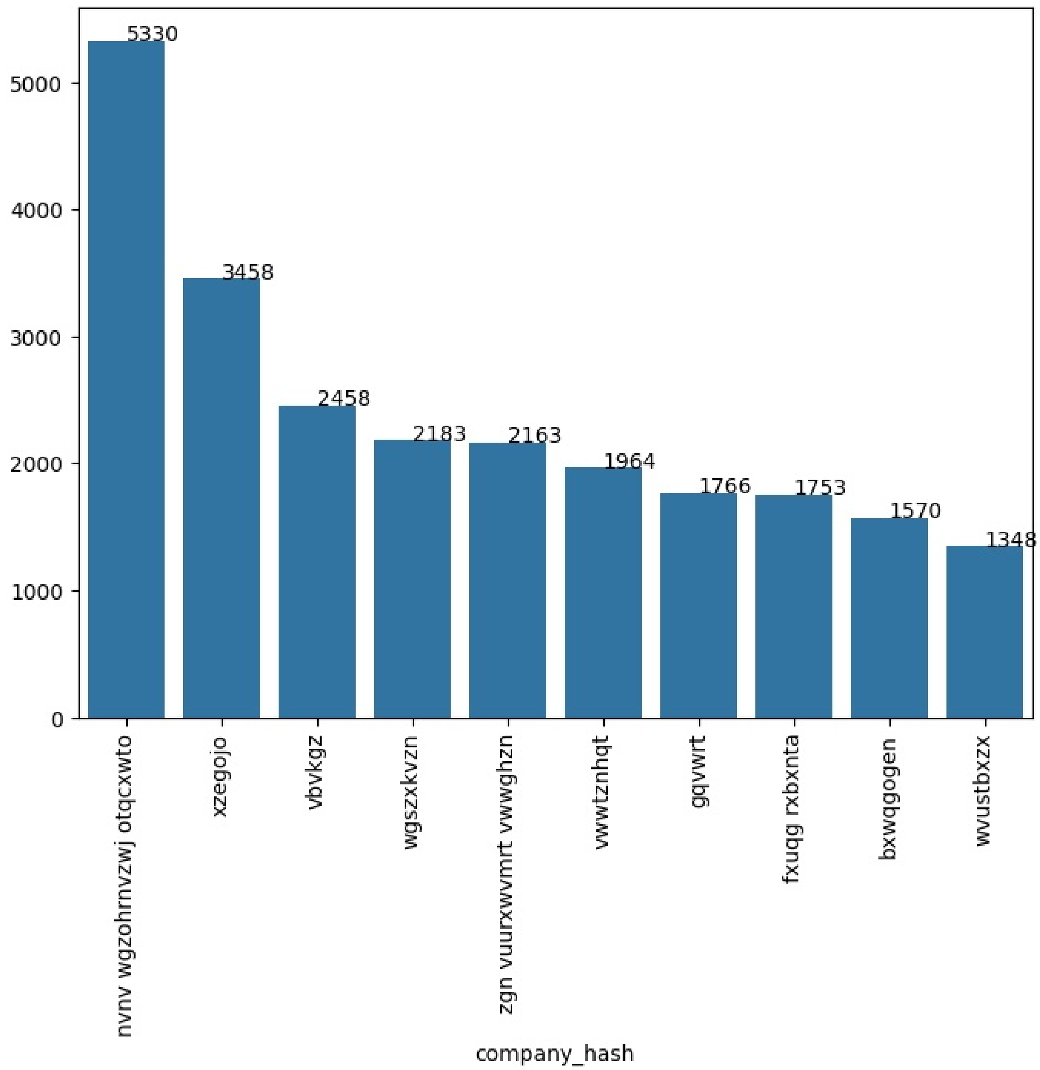
	orgyear	count
0	2016.0	17332
1	2018.0	16828
2	2017.0	16557
3	2015.0	15982
4	2019.0	15000
5	2014.0	13281
6	2013.0	10013
7	2020.0	9180
8	2012.0	8555
9	2011.0	6446

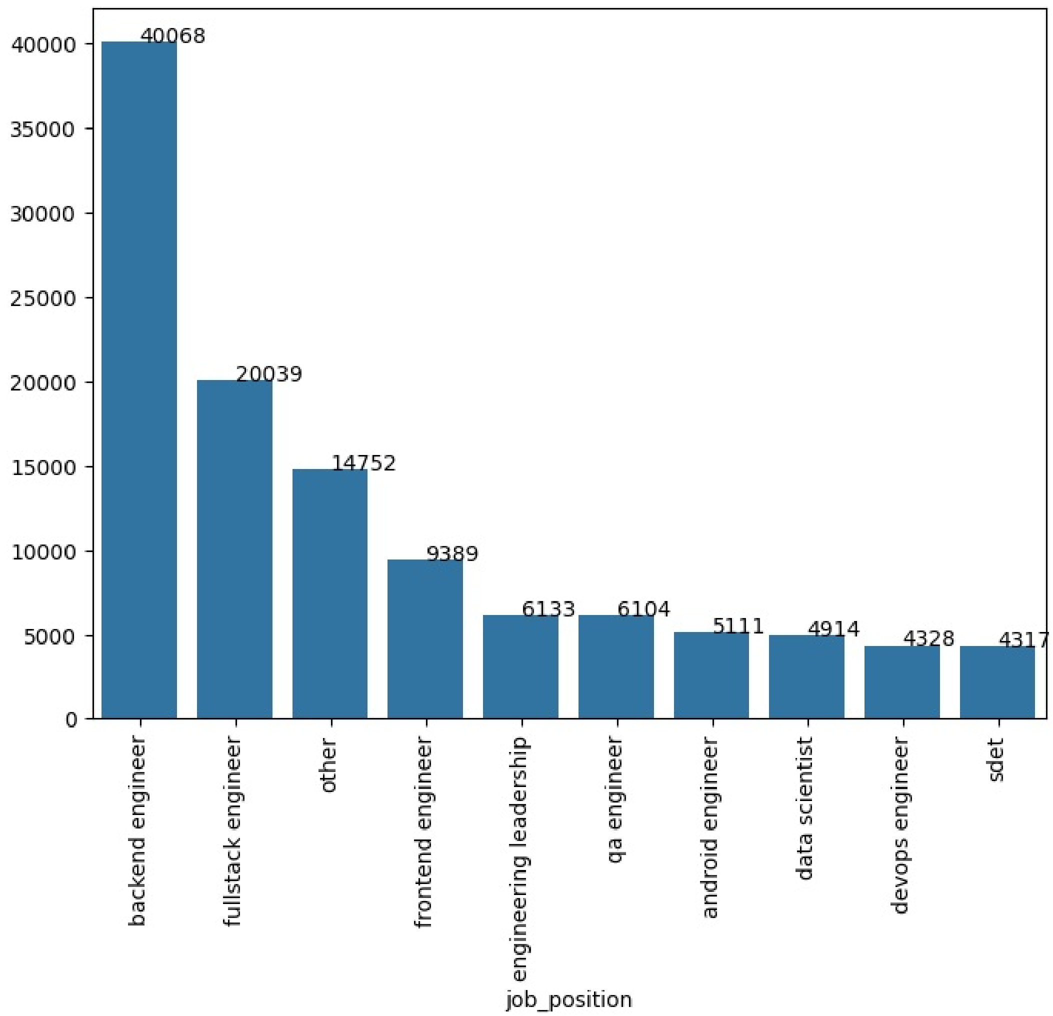
	ctc_updated_year	count
0	2019.0	53188
1	2021.0	42121
2	2020.0	36943
3	2017.0	7025
4	2018.0	6464
5	2016.0	4993
6	2015.0	2551

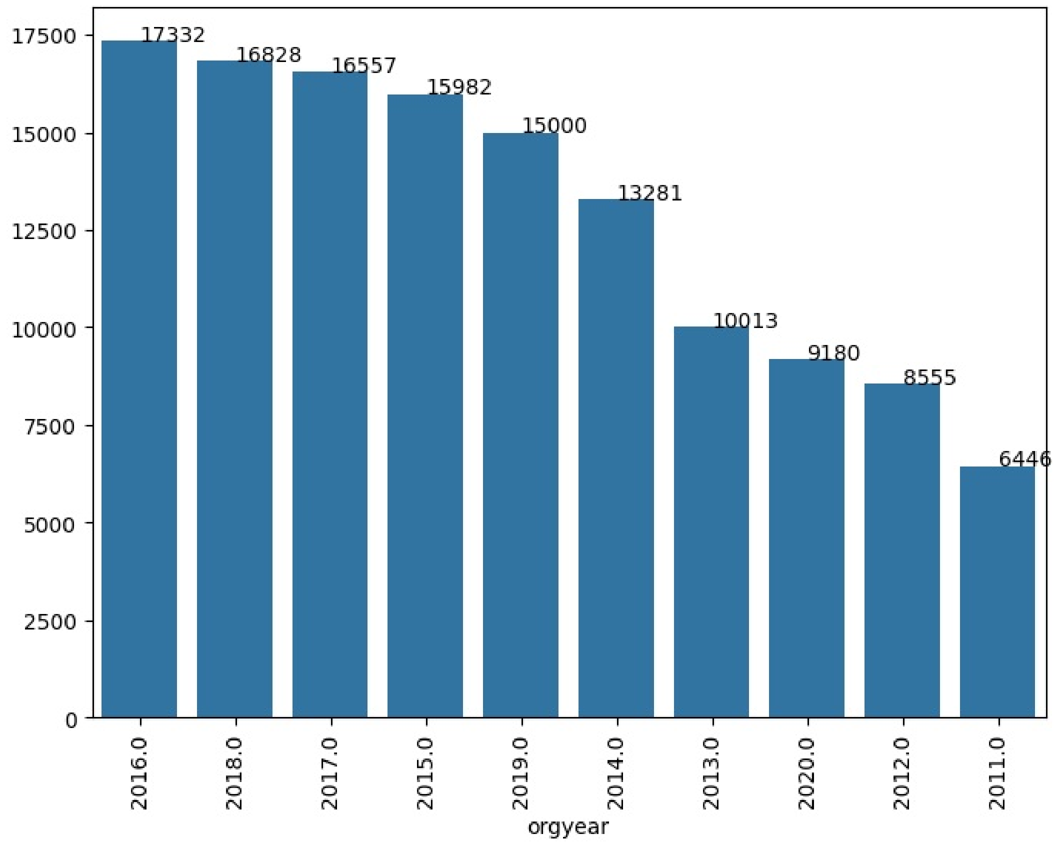
	Income_bin	count
0	Low	76213
1	Medium	61763
2	High	15309

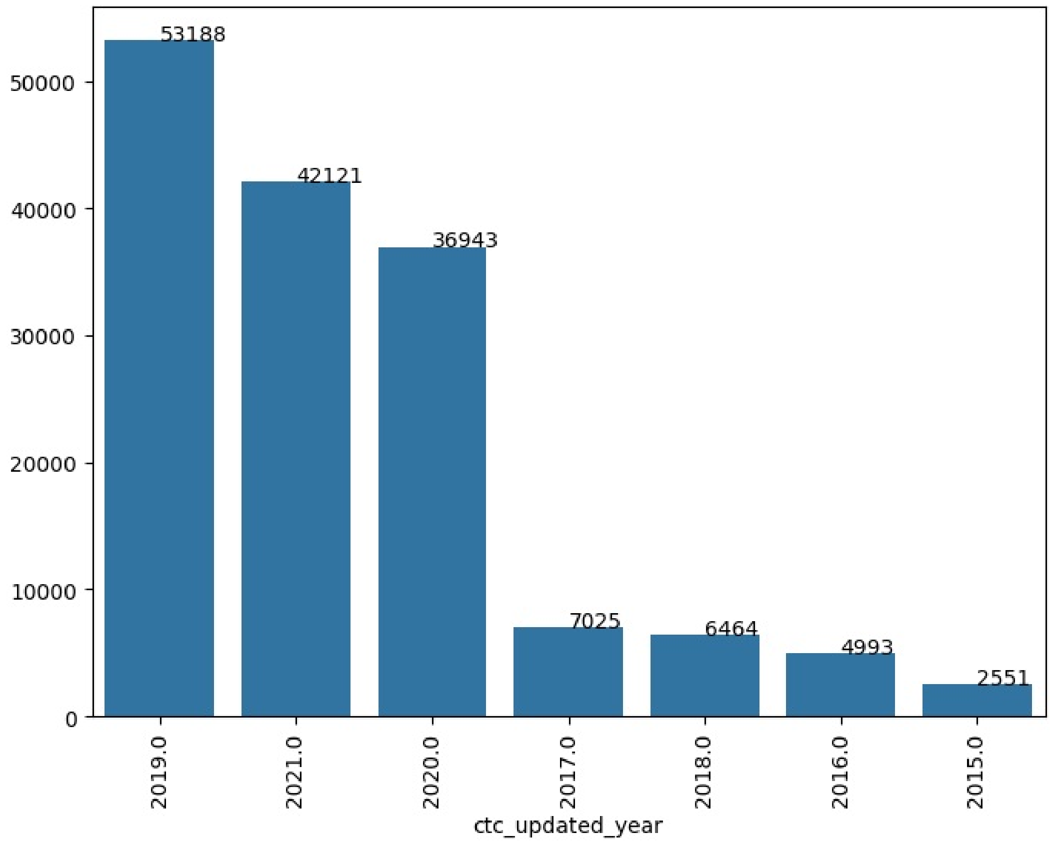
	Job_prominance	count
0	Medium_prominant	99638
1	Highly_prominant	28302
2	Less_prominant	5162

	current_year_ctc_updated	count
0	0	111164
1	1	42121

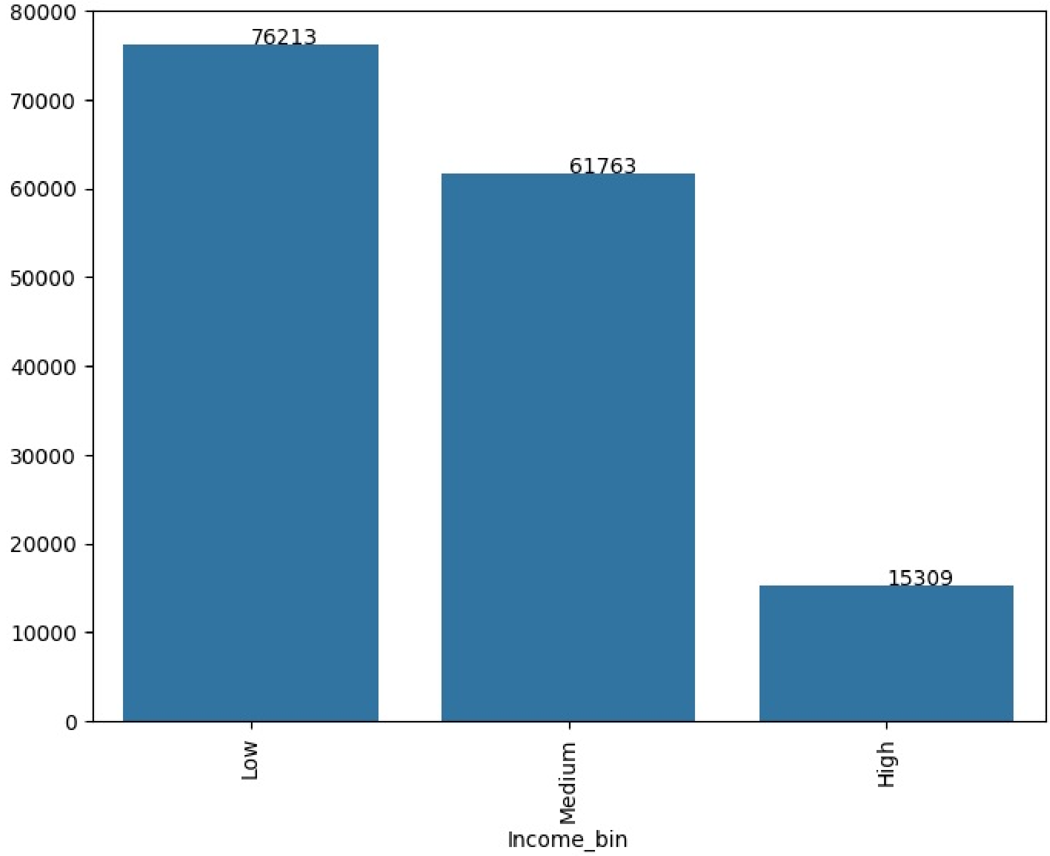



40000
35000
30000
25000
20000
15000
10000
5000
0
40068
20039
14752
9389
6133
6104
5111
4914
4328
4317
backend engineer
fullstack engineer
other
frontend engineer
engineering leadership
qa engineer
android engineer
data scientist
devops engineer
sdet
job_position

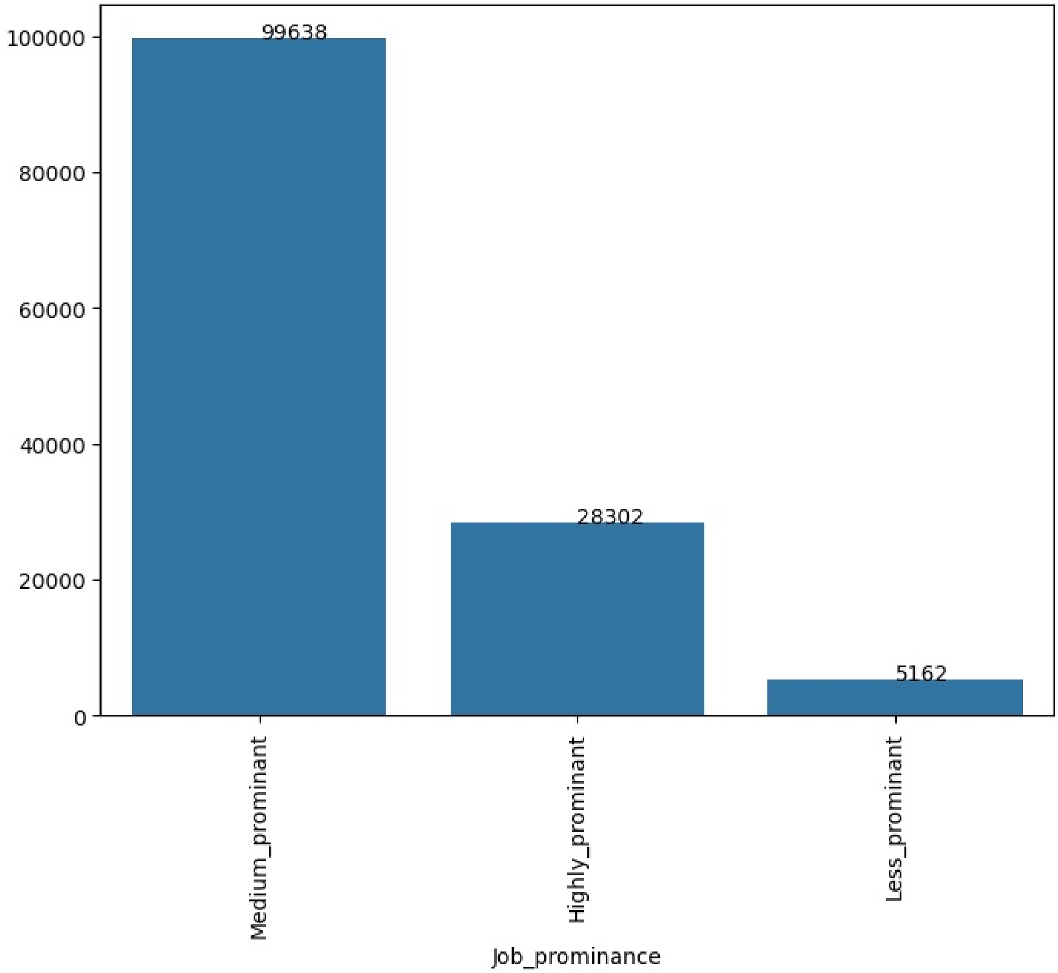

17500
15000
12500
10000
7500
5000
2500
0
17332
16828
16557
15982
15000
13281
10013
9180
8555
6446
2016.0
2018.0
2017.0
2015.0
2019.0
2014.0
2013.0
2020.0
2012.0
2011.0
orgyear

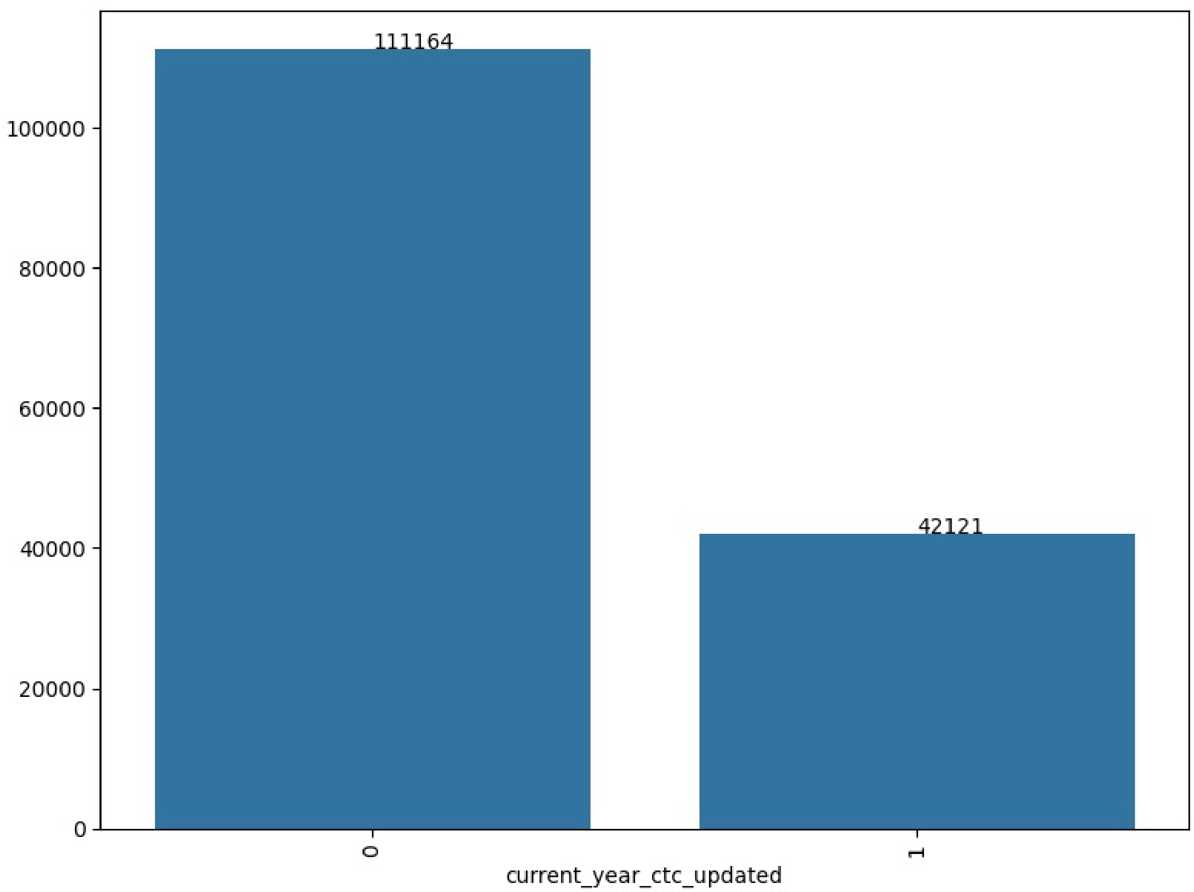


53188
42121
36943
7025
6464
4993
2551
0
10000
20000
30000
40000
50000
2019.0
2021.0
2020.0
2017.0
2018.0
2016.0
2015.0
ctc_updated_year



76213
61763
15309
0
10000
20000
30000
40000
50000
60000
70000
80000
Low
Medium
High
Income_bin


99638
28302
5162
100000
80000
60000
40000
20000
0
Medium_prominant
Highly_prominant
Less_prominant
Job_prominance



Analysis of categorical columns

- The dataset has 153443 unique learners.

- Majority of the learners (22%) are employed at the company 'nvnv wgzohrnvzwj otqcxwto' while 14% work at 'xzegojo' and 10% at 'vbvkgz'.

- 35% of students are having the current job position as Backend Engineering followed by 16% who are Fullstack Engineers.

- The joining year of 13.5% employees at their current company is 2016 closely followed by 2018 an 2017.

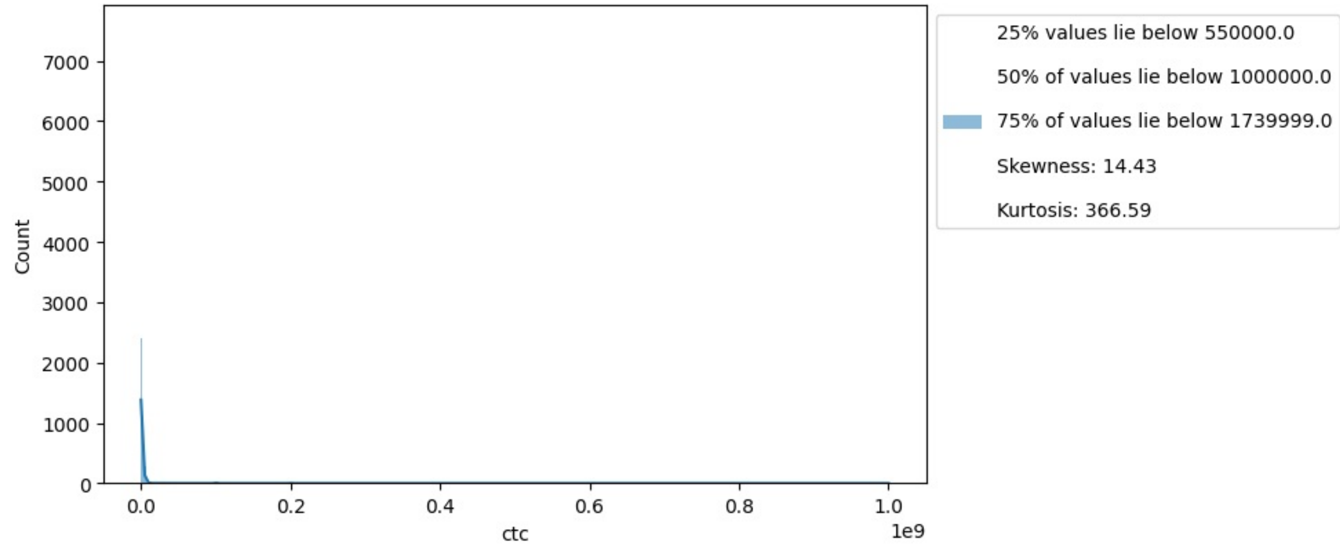
- Amost 35% of learners has their CTC updated in the year 2019 and 27% and 24% got their ctc updated in 2021 and 2020 respectively. A total of only 13% students had their ctc updated before the year 2018.

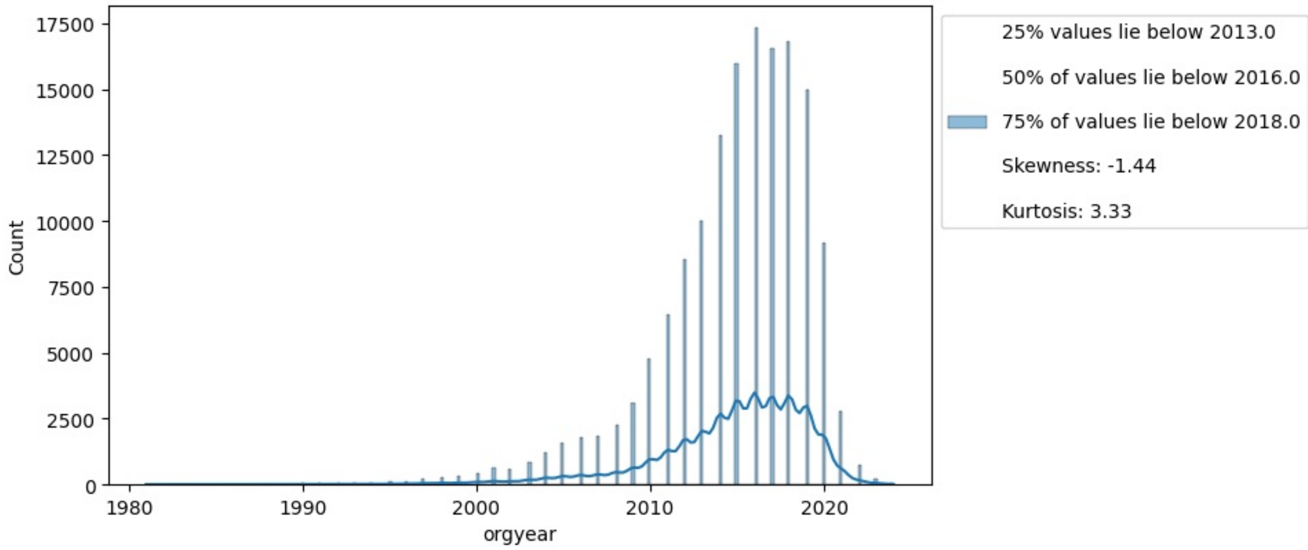
- 50% students are of low income bin, 40% in medium and only 10% learners are of high income bin.

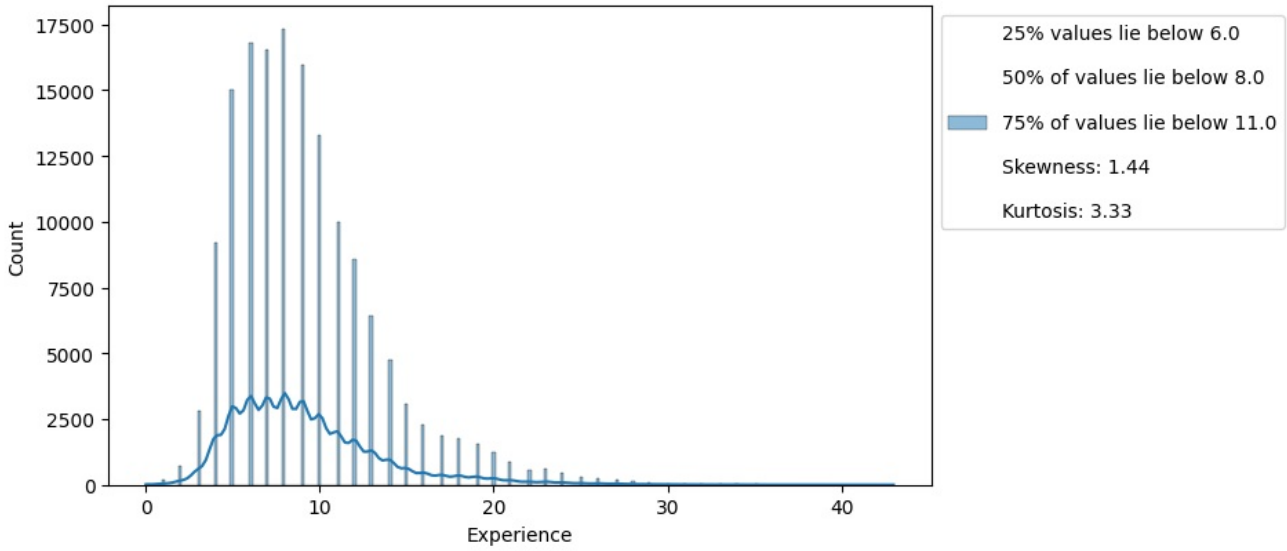
- 74% employees are in job_positions having medium prominance

b. Numerical columns analysis.

In [30]:
```
from scipy.stats import skew, kurtosis


hist_cols=[ 'ctc',  'orgyear', 'Experience']
n=1
plt.figure(figsize=(10,20))
for i in hist_cols:

    plt.subplot(5,1,n)

    sns.histplot(data=df_agg.dropna(subset=[i]), x=i, kde=True,
             label=f'25% values lie below {round(np.percentile(df_agg[i].dropna(), 25), 2)}\n\n'
                   f'50% of values lie below {round(np.percentile(df_agg[i].dropna(), 50), 2)}\n\n'
                   f'75% of values lie below {round(np.percentile(df_agg[i].dropna(), 75), 2)}\n\n'
                   f'Skewness: {np.round(skew(df_agg[i].dropna()), 2)}\n\n'
                   f'Kurtosis: {round(kurtosis(df_agg[i].dropna()), 2)}')

    plt.legend(loc='upper left', bbox_to_anchor=(1, 1))
    n+=1


plt.tight_layout()
plt.show()
```



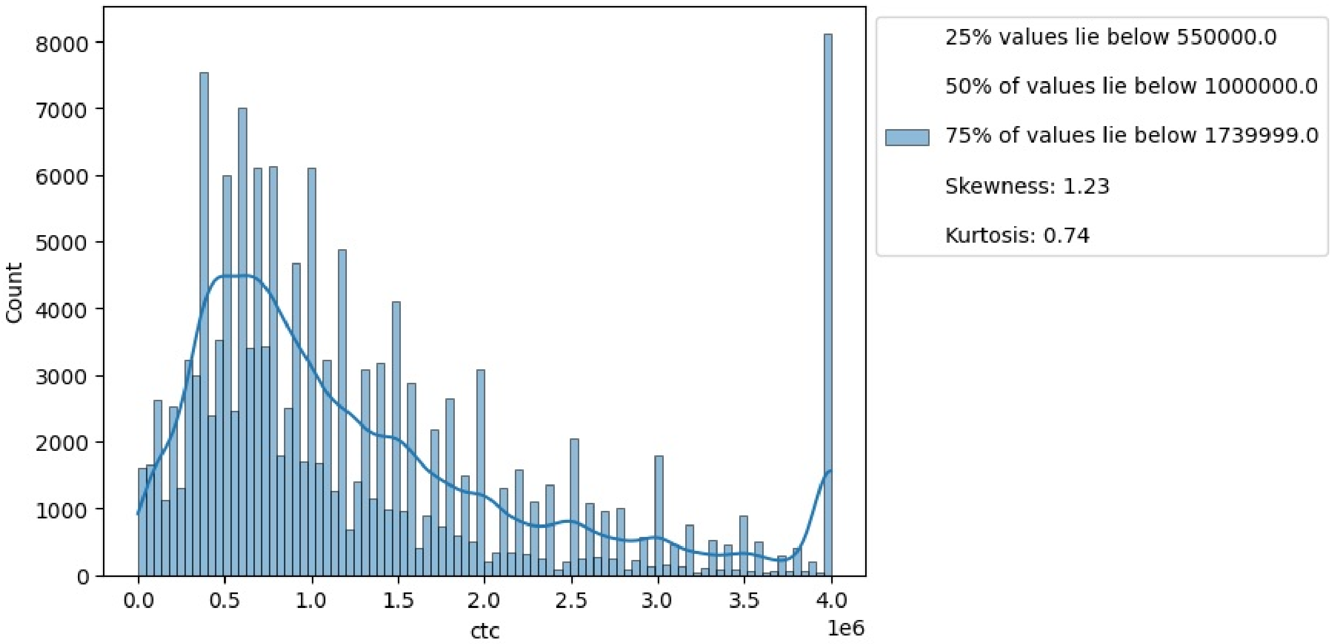




In [31]:

```
ctc_clipped=np.clip(df_agg['ctc'],np.percentile(df_agg['ctc'].dropna(),0),np.percentile(df_agg['ctc'].dropna(),
plt.plot(figsize=(6,6))
#sns.histplot(data=df_agg,x=i,kde=True,label=f'25% values lie below {round(np.percentile(df_agg[i],25),2)}\n\n

sns.histplot(data=ctc_clipped, kde=True,
         label=f'25% values lie below {round(np.percentile(ctc_clipped.dropna(), 25), 2)}\n\n'
               f'50% of values lie below {round(np.percentile(ctc_clipped.dropna(), 50), 2)}\n\n'
               f'75% of values lie below {round(np.percentile(ctc_clipped.dropna(), 75), 2)}\n\n'
               f'Skewness: {np.round(skew(ctc_clipped.dropna()), 2)}\n\n'
               f'Kurtosis: {round(kurtosis(ctc_clipped.dropna()), 2)}')

plt.legend(loc='upper left', bbox_to_anchor=(1, 1))
```

Out[31]: <matplotlib.legend.Legend at 0x2903452f710>



In [32]:
```
df_agg['Experience'].describe()
```

Out[32]:
```
count    153285.000000
mean          9.191891
std           4.357146
min           0.000000
25%           6.000000
50%           8.000000
75%          11.000000
max          43.000000
Name: Experience, dtype: float64
```

<u>Analysis of numerical columns</u>

- It is clear from the plot that the column ctc is extremely right-skewed. This means that the majority of the data points are concentrated on the left which are lower values , but there are a few outliers or extreme values on the right. The column in also having very high value for kurtosis which means it has heavier tails and a sharper peak around the mean compared to a normal distribution.

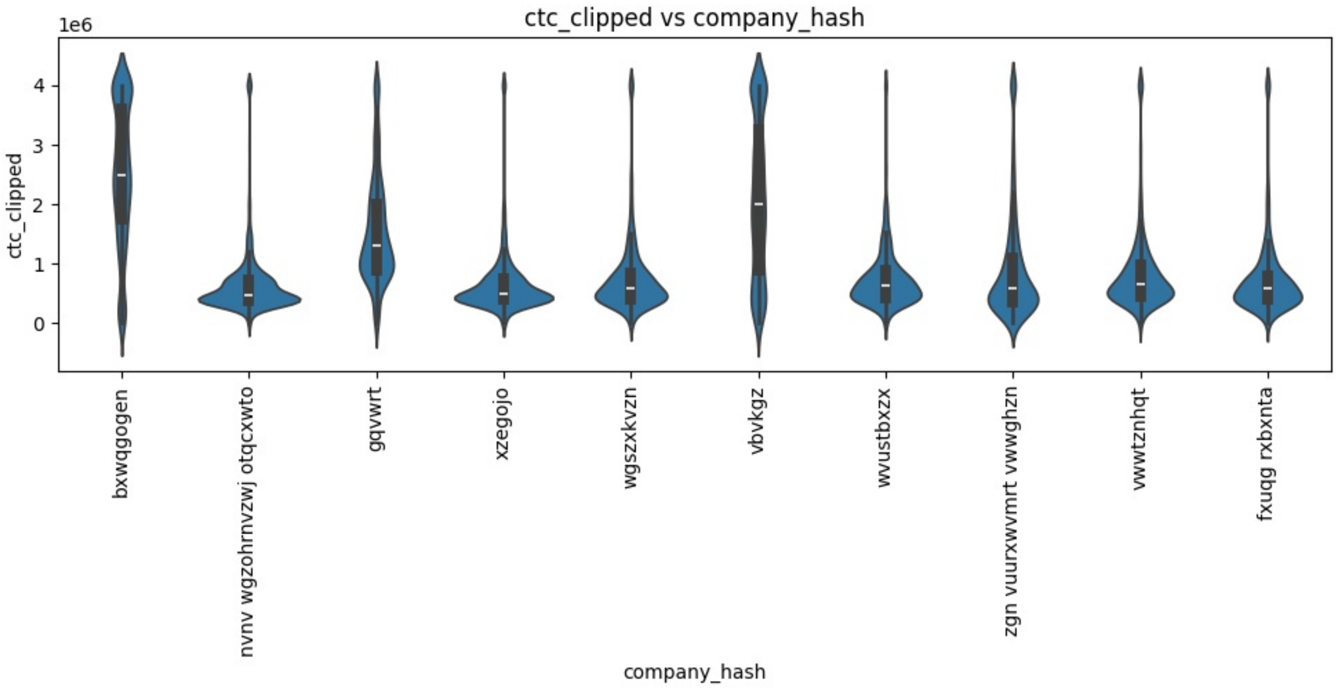
- The values in the column range from 2 to 1.0e+9. Upon removing the outliers in column ctc, we can see 75% of values lie before 1739999.

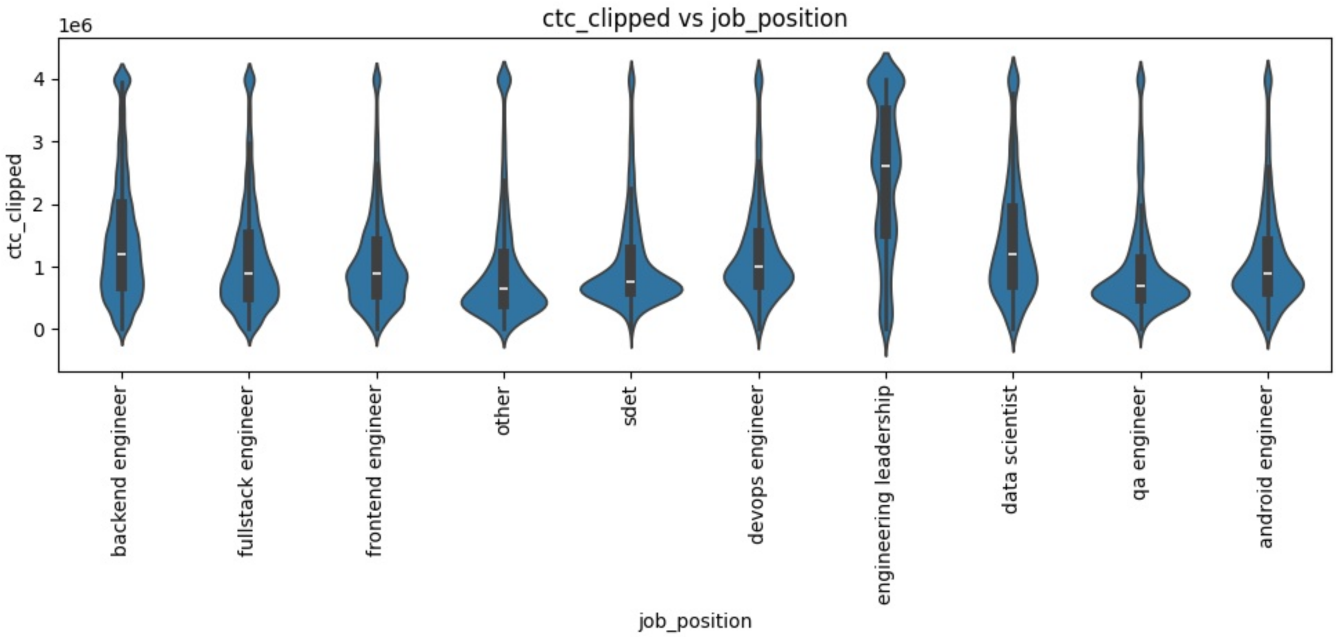
- The column orgyear is left skewed with leptokurtic distribution. The values in the column ranges from 1970 to 2024 where 87% of employees joined their current company between 2010 and 2020.

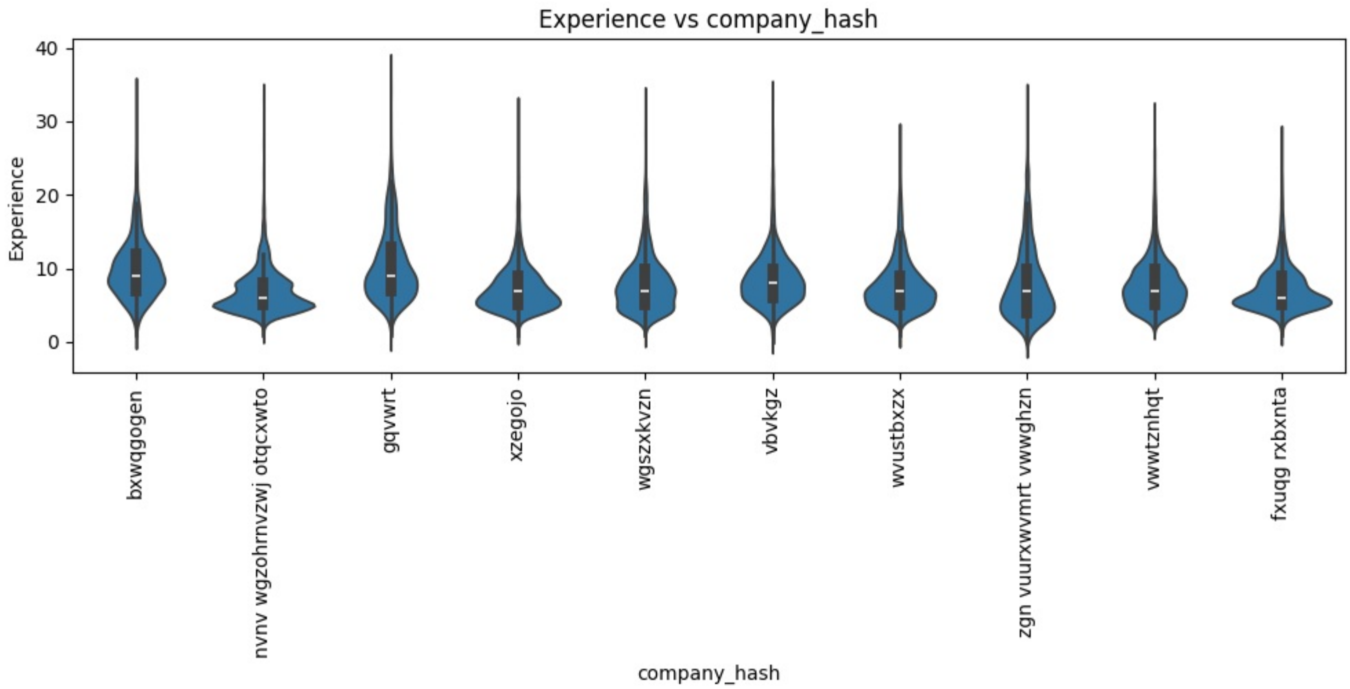
- Experience of employees ranges from 0 to 54 years where the data is rght skewed and leptokurtic. 75% of employees are hving experience of below 11 years.

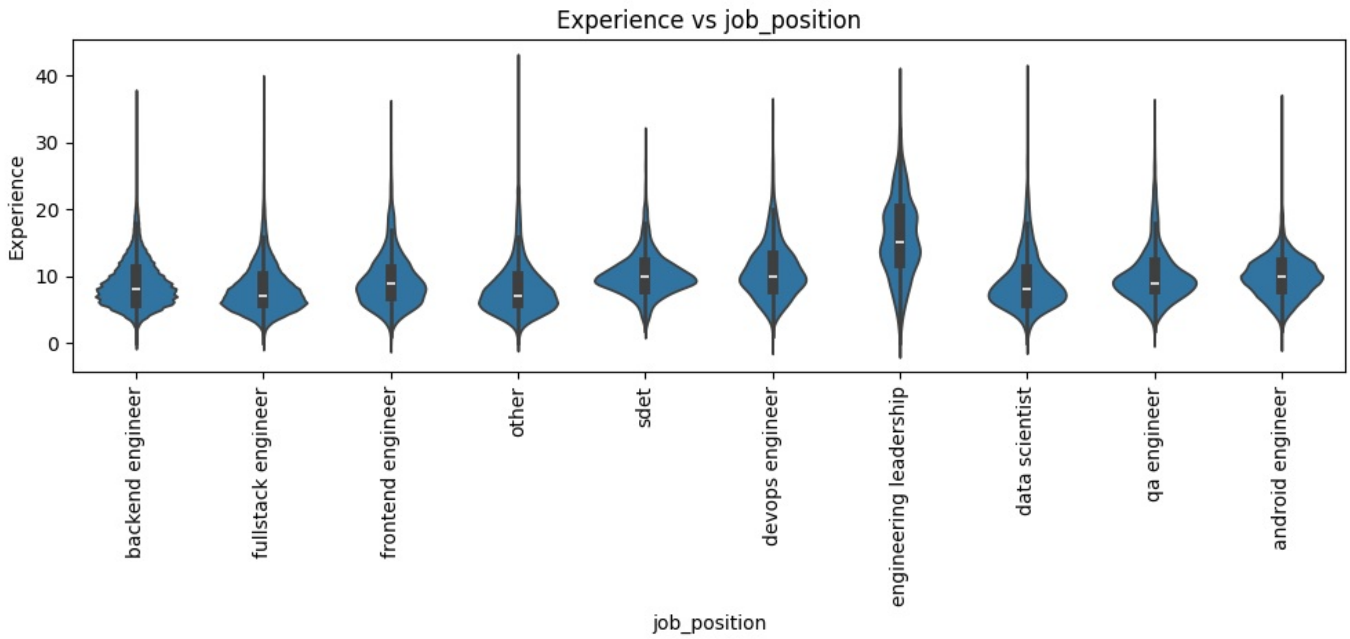
In [33]:

```
df_agg['ctc_clipped']=np.clip(df_agg['ctc'],np.percentile(df_agg['ctc'].dropna(),0),np.percentile(df_agg['ctc']
top_comp=df_agg.groupby('company_hash').size().sort_values(ascending=False).head(10).index
top_job=df_agg.groupby('job_position').size().sort_values(ascending=False).head(10).index

viol_nums=['ctc_clipped','Experience']
viol_cat=[top_comp,top_job]
viol_cat_cols=['company_hash', 'job_position']

n=1
plt.figure(figsize=(10, 20))

for i in viol_nums:
    for j in range(len(viol_cat)):
        filtered_df=df_agg[df_agg[viol_cat_cols[j]].isin(viol_cat[j])]
        plt.subplot(4, 1, n)
        sns.violinplot(x=filtered_df[viol_cat_cols[j]],y=filtered_df[i])
        plt.title(f'{i} vs {viol_cat_cols[j]}')
        n += 1

        plt.xticks(rotation=90)

plt.tight_layout()
plt.show()
```



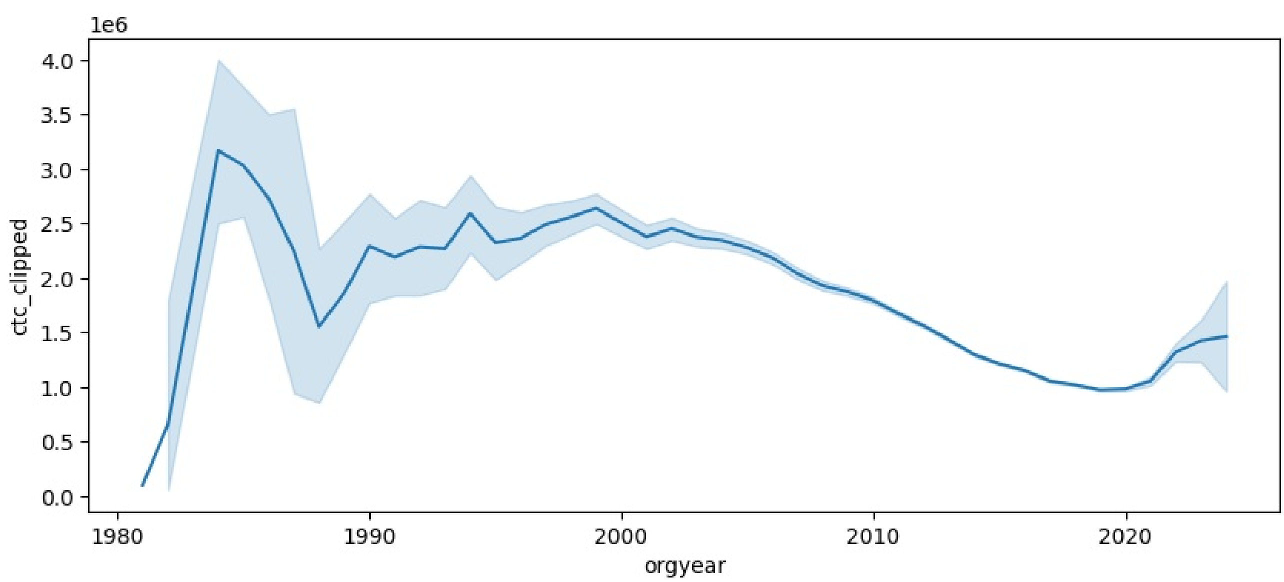






In [34]:
```
plt.figure(figsize=(10,4))
sns.lineplot(data=df_agg,x='orgyear',y='ctc_clipped')
```

Out[34]: <Axes: xlabel='orgyear', ylabel='ctc_clipped'>



The top 10 employers and job positions are taken for analysis of ctc clipped value and Experience.

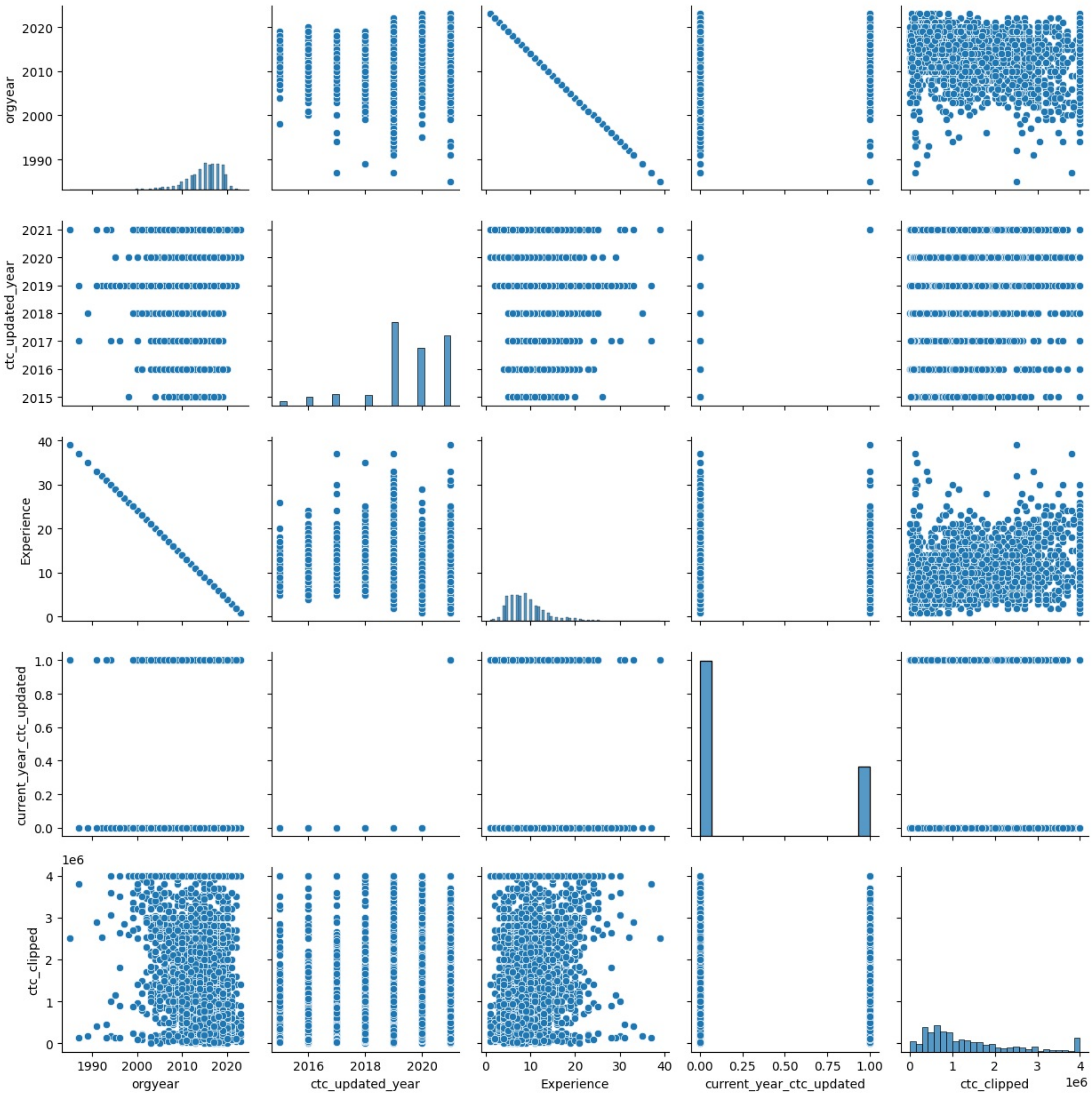
- Most companies give a ctc of below 2000000 to the learners. Companies like 'bxwqgogen' and 'vbvkgz' are seen to give higher ctc to more employees.

- Higher ctc is given to employees in Engineering Leadership position. All other job positions are given an average ctc of below 3000000.

- Most learners from all companies are showing an experience of below 20 years. Here learners in Engineering Leadership position seems to have more experience compared to others.

In [35]:

```
sns.pairplot(df_agg.sample(5000,random_state=30).iloc[:,2:])
plt.tight_layout()
plt.show()
```



In [36]:
```
plt.figure(figsize=(8,6))
sns.heatmap(df_agg.iloc[:,2:].corr(), annot=True,fmt='.2f')
plt.tight_layout()
plt.show()
```

```
C:\Users\devip\AppData\Local\Temp\ipykernel_20724\2486540480.py:2: FutureWarning: The default value of numeric_o
nly in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or
specify the value of numeric_only to silence this warning.
  sns.heatmap(df_agg.iloc[:,2:].corr(), annot=True,fmt='.2f')
```

- Experience and orgyear are having high negative correlation as experience increases when the learner has joined the company earlier.

- There is a positive correlation between ctc and Experience. With increase in experience, employee can demand more ctc.

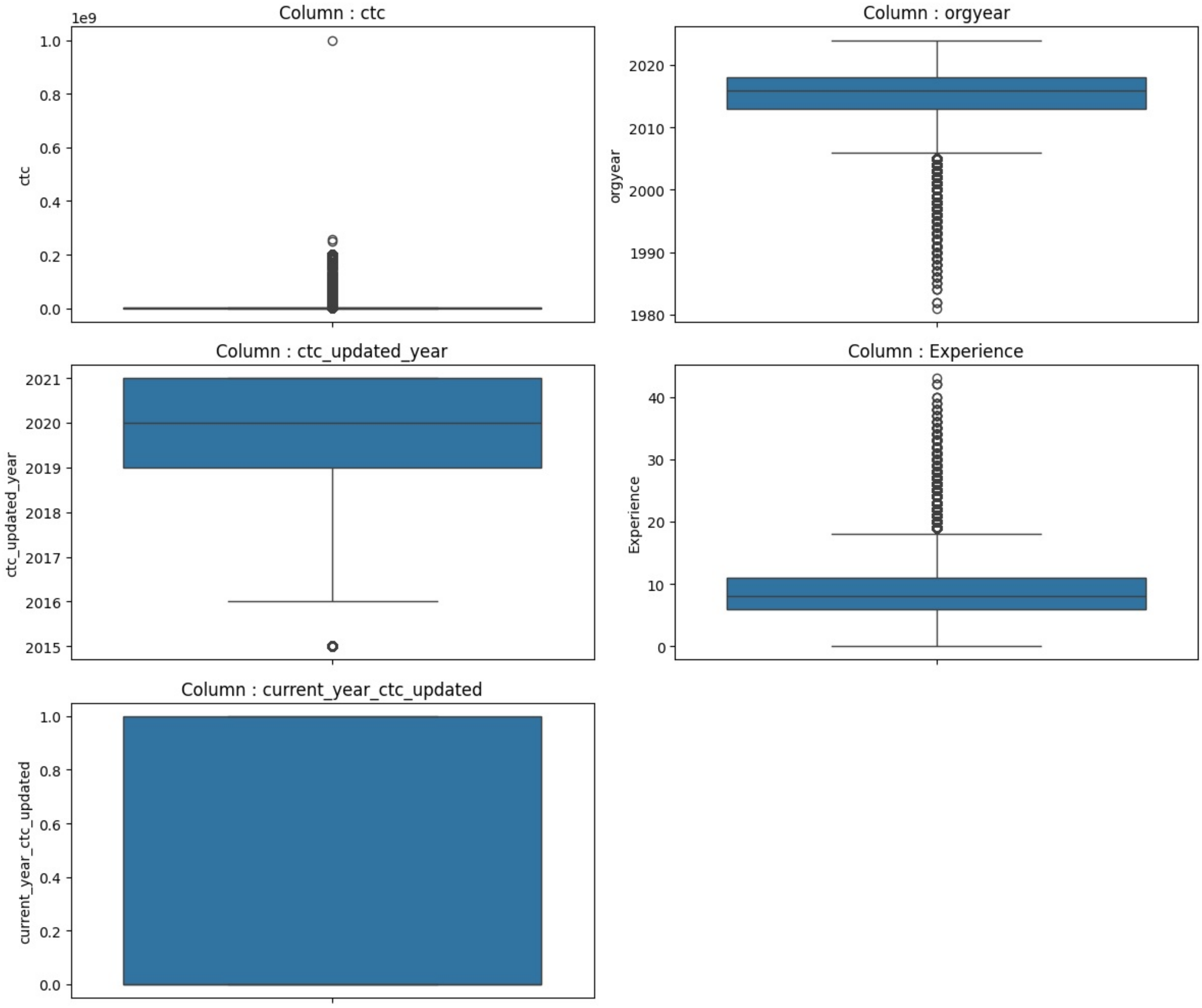
- The negative correlation between experience and ctc is because the earlier the employee joins the company, the salary will be more.

In [37]:
```
df_agg.drop('ctc_clipped',axis=1,inplace=True)
```

5. Detecting and treating outliers in the data.

In [38]:

```
df_agg.columns=df_agg.columns.str.strip()
num_cols=df_agg.select_dtypes(include=['number']).columns


no=1
plt.figure(figsize=(12,20))
for col in num_cols:
    plt.subplot(6,2,no)
    plt.title('Column : ' +col)
    sns.boxplot(y=df_agg[col])
    no+=1
plt.tight_layout()
plt.show()
```



In [39]:

```
total=len(df_agg)

for col in num_cols:

    q1=np.percentile(df_agg[col].dropna(),25)
    q3=np.percentile(df_agg[col].dropna(),75)
    l_limit=q1-(1.5*(q3-q1))
    u_limit=q3+(1.5*(q3-q1))

    count=len(df_agg[(df_agg[col]>u_limit) | (df_agg[col]<l_limit)])

    print(f"Column: {col}\nlower_limit: {l_limit}\nupper_limit : {u_limit}\nPercentage of outliers: {round((cou
```

```
Column: ctc
lower_limit: -1234998.5
upper_limit : 3524997.5
Percentage of outliers: 6.43%


Column: orgyear
lower_limit: 2005.5
upper_limit : 2025.5
Percentage of outliers: 4.28%


Column: ctc_updated_year
lower_limit: 2016.0
upper_limit : 2024.0
Percentage of outliers: 1.66%


Column: Experience
lower_limit: -1.5
upper_limit : 18.5
Percentage of outliers: 4.28%


Column: current_year_ctc_updated
lower_limit: -1.5
upper_limit : 2.5
Percentage of outliers: 0.0%
```

All numerical columns are having outliers with ctc having 6.4% outlier data. Clipping the outliers

In [40]:
```
# df_agg['ctc']=np.clip(df_agg['ctc'],np.percentile(df_agg['ctc'].dropna(),0),np.percentile(df_agg['ctc'].dropn
# df_agg['Experience']=np.clip(df_agg['Experience'],np.percentile(df_agg['Experience'],0),np.percentile(df_agg[
# df_agg['ctc_updated_year']=np.clip(df_agg['ctc_updated_year'],np.percentile(df_agg['ctc_updated_year'],5),np.
# df_agg['orgyear']=np.clip(df_agg['orgyear'],np.percentile(df_agg['orgyear'].dropna(),5),np.percentile(df_agg[
```

In [41]:
```
for col in num_cols:

    q1=np.percentile(df_agg[col].dropna(),25)
    q3=np.percentile(df_agg[col].dropna(),75)
    l_limit=q1-(1.5*(q3-q1))
    u_limit=q3+(1.5*(q3-q1))

    df_agg[col]=np.clip(df_agg[col],l_limit,u_limit)
```

In [42]:
```
total=len(df_agg)

for col in num_cols:

    q1=np.percentile(df_agg[col].dropna(),25)
    q3=np.percentile(df_agg[col].dropna(),75)
    l_limit=q1-(1.5*(q3-q1))
    u_limit=q3+(1.5*(q3-q1))

    count=len(df_agg[(df_agg[col]>u_limit) | (df_agg[col]<l_limit)])

    print(f"Column: {col}\nlower_limit: {l_limit}\nupper_limit : {u_limit}\nPercentage of outliers: {round((cou
```

```
Column: ctc
lower_limit: -1234998.5
upper_limit : 3524997.5
Percentage of outliers: 0.0%


Column: orgyear
lower_limit: 2005.5
upper_limit : 2025.5
Percentage of outliers: 0.0%


Column: ctc_updated_year
lower_limit: 2016.0
upper_limit : 2024.0
Percentage of outliers: 0.0%


Column: Experience
lower_limit: -1.5
upper_limit : 18.5
Percentage of outliers: 0.0%


Column: current_year_ctc_updated
lower_limit: -1.5
upper_limit : 2.5
Percentage of outliers: 0.0%
```

---

6. Encoding categorical values.

In [43]:
```
df_agg.select_dtypes(include=['object','datetime'])
```

Out[43]:

	email_hash	company_hash	job_position	Income_bin	Job_prominance
0	00003288036a44374976948c327f246fdbdf0778546904...	bxwqgogen	backend engineer	High	Medium_prominant
1	0000aaa0e6b61f7636af1954b43d294484cd151c9b3cf6...	nqsn axsxnvr	backend engineer	Low	Medium_prominant
2	0000d58fbc18012bf6fa2605a7b0357d126ee69bc41032...	gunhb	fullstack engineer	Medium	Medium_prominant
3	000120d0c8aa304fcf12ab4b85e21feb80a342cfea03d4...	bxwqgotbx wgqugqvnxgz	fullstack engineer	Medium	Medium_prominant
4	00014d71a389170e668ba96ae8e1f9d991591acc899025...	fvrbvqn rvmo	NaN	High	NaN
...	...	...	...	...	...
153438	fffc254e627e4bd1bc0ed7f01f9aebbba7c3cc56ac914e...	tqxwoogz ogenfvqt wvbuho	qa engineer	High	Medium_prominant
153439	fffcf97db1e9c13898f4eb4cd1c2fe862358480e104535...	trnqvcg	NaN	Medium	NaN
153440	fffe7552892f8ca5fb8647d49ca805b72ea0e9538b6b01...	znn avnv srgmvr atrxctqj otqcxwto	devops engineer	Low	Medium_prominant
153441	ffff49f963e4493d8bbc7cc15365423d84a767259f7200...	zwq wgqugqvnxgz	fullstack engineer	Low	Medium_prominant
153442	ffffa3eb3575f43b86d986911463dce7bcadcea227e5a4...	sgrabvz ovwyo	fullstack engineer	Medium	Medium_prominant

153285 rows × 5 columns

In [44]:
```
df_agg_processed=df_agg.copy(deep=True)
```

Dropping column email_hash as it is a unique identifier.

In [45]:
```
df_agg_processed.drop('email_hash',axis=1,inplace=True)
```

Implementing Target Encoding for company_hash and job_position.

In [46]:
```
import category_encoders as ce


target_enc = ce.TargetEncoder(cols=['job_position', 'company_hash'], smoothing=0.3)
df_agg_processed[['job_position_encoded', 'company_hash_encoded']] = target_enc.fit_transform(df_agg[['job_posi
```

In [47]:
```
df_agg_processed.head()
```

Out[47]:

	ctc	company_hash	job_position	orgyear	ctc_updated_year	Experience	Income_bin	Job_prominance	current_year_ct
0	3500000.0	bxwqgogen	backend engineer	2012.0	2019.0	12.0	High	Medium_prominant	
1	250000.0	nqsn axsxnvr	backend engineer	2013.0	2020.0	11.0	Low	Medium_prominant	
2	1300000.0	gunhb	fullstack engineer	2021.0	2019.0	3.0	Medium	Medium_prominant	
3	2000000.0	bxwqgotbx wgqugqvnxgz	fullstack engineer	2005.5	2021.0	18.5	Medium	Medium_prominant	
4	3400000.0	fvrbvqn rvmo	NaN	2009.0	2018.0	15.0	High	NaN	

In [48]:
```
df_agg_processed.drop(['job_position','company_hash'],axis=1,inplace=True)
```

Dropping columns Income bin,job prominance and current_year_ctc_updated as it may generalise the data.

In [49]:
```
df_agg_processed.drop(['Income_bin','Job_prominance' , 'current_year_ctc_updated'],axis=1,inplace=True)
```

In [50]:
```
df_agg_processed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 153285 entries, 0 to 153442
Data columns (total 6 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   ctc                   153285 non-null  float64
 1   orgyear               153285 non-null  float64
 2   ctc_updated_year      153285 non-null  float64
 3   Experience            153285 non-null  float64
 4   job_position_encoded  153285 non-null  float64
 5   company_hash_encoded  153285 non-null  float64
dtypes: float64(6)
memory usage: 8.2 MB
```

7. Scaling data and Filling missing values.

Columns 'job_position' is having 25% missing values. 'orgyear' and 'company_hash' are also having null values. Using Standard Scalar to scale the data and KNN Imputer to fill the missing values.

In [51]:
```
from sklearn.impute import KNNImputer
from sklearn.preprocessing import StandardScaler



num_cols = df_agg_processed.select_dtypes(include=[np.number])


stdscaler = StandardScaler()
num_cols_std = stdscaler.fit_transform(num_cols)


imputer = KNNImputer(n_neighbors=5)
num_cols_imp = imputer.fit_transform(num_cols_std)


#num_cols_imp= stdscaler.inverse_transform(num_cols_imp)
num_cols_imp_df = pd.DataFrame(num_cols_imp, columns=num_cols.columns)

df_agg_processed = num_cols_imp_df

df_agg_processed.head()
```

Out[51]:

	ctc	orgyear	ctc_updated_year	Experience	job_position_encoded	company_hash_encoded
0	2.316688	-0.767619	-0.384918	0.767619	0.513360	2.564056
1	-1.066698	-0.508776	0.371141	0.508776	0.513360	-0.107703
2	0.026396	1.561971	-0.384918	-1.561971	-0.396470	0.295069
3	0.755125	-2.450101	1.127200	2.450101	-0.396470	-0.107703
4	2.212584	-1.544149	-1.140976	1.544149	-0.134507	2.510274

In [52]:
```
(df_agg_processed.isna().sum(axis=0)/len(df_agg_processed))*100
```

Out[52]:
```
ctc                     0.0
orgyear                 0.0
ctc_updated_year        0.0
Experience              0.0
job_position_encoded    0.0
company_hash_encoded    0.0
dtype: float64
```

8. Data Preprocessing.

a. Train-test split.

In [53]:
```
from sklearn.model_selection import train_test_split

X_train, X_test = train_test_split(df_agg_processed, test_size=0.2, random_state=42)
```

In [54]:
```
X_train.shape, X_test.shape
```

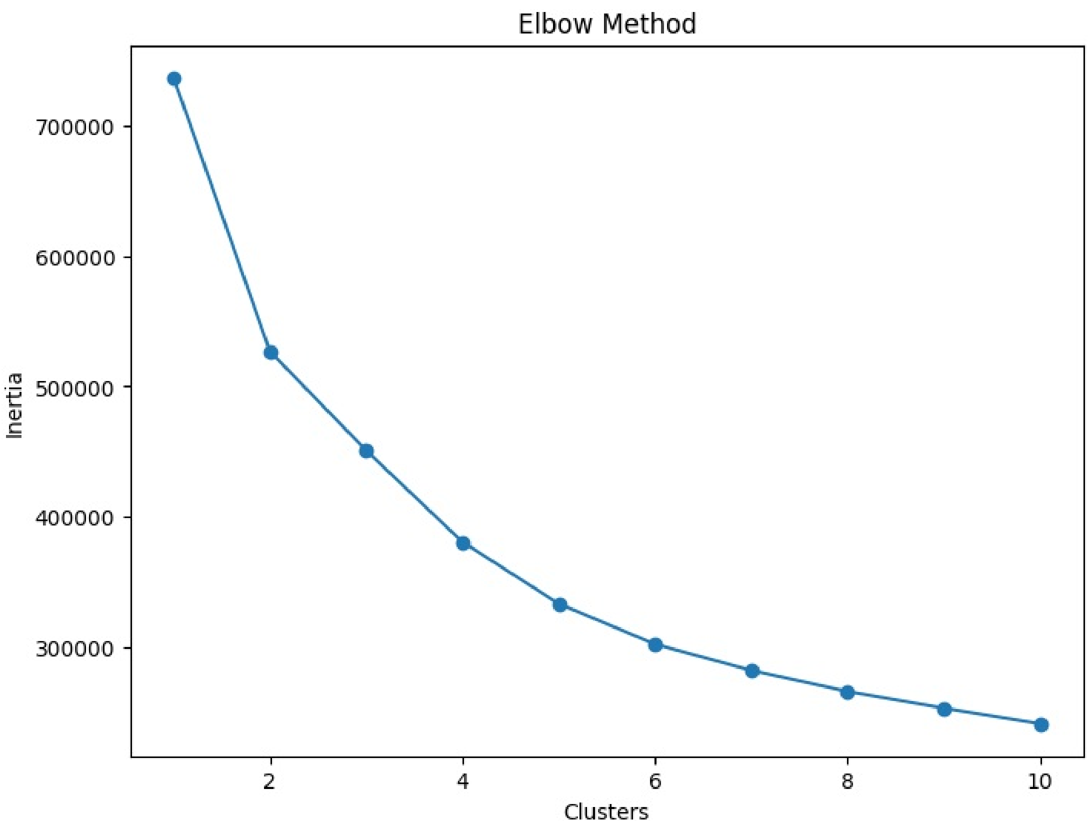
Out[54]:
```
((122628, 6), (30657, 6))
```

In [55]:
```
Xtrain_copy=X_train.copy(deep=True)
```

---

9. K means clustering

In [56]:
```
from sklearn.cluster import KMeans

result=[]

for k in range(1,11):
    kmeans=KMeans(n_clusters=k,random_state=42,n_init=10)
    kmeans.fit(X_train)
    result.append(kmeans.inertia_)

plt.figure(figsize=(8,6))
plt.plot(range(1,11),result,marker='o')
plt.xlabel('Clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method')
plt.show()
```



Elbow method is used to assess the optimum number of clusters. Here at 3 or 4 clusters, inertia starts to slow down So this may be the optimum number.

In [57]:
```
kmeans = KMeans(n_clusters=4, random_state=42, n_init=10)
kmeans.fit(X_train)
```

Out[57]:
```
KMeans
KMeans(n_clusters=4, n_init=10, random_state=42)
```

In [58]:
```
kmlabels = kmeans.labels_
inertia = kmeans.inertia_
print(f"Inertia : {inertia}")
```

```
Inertia : 380400.65121601307
```

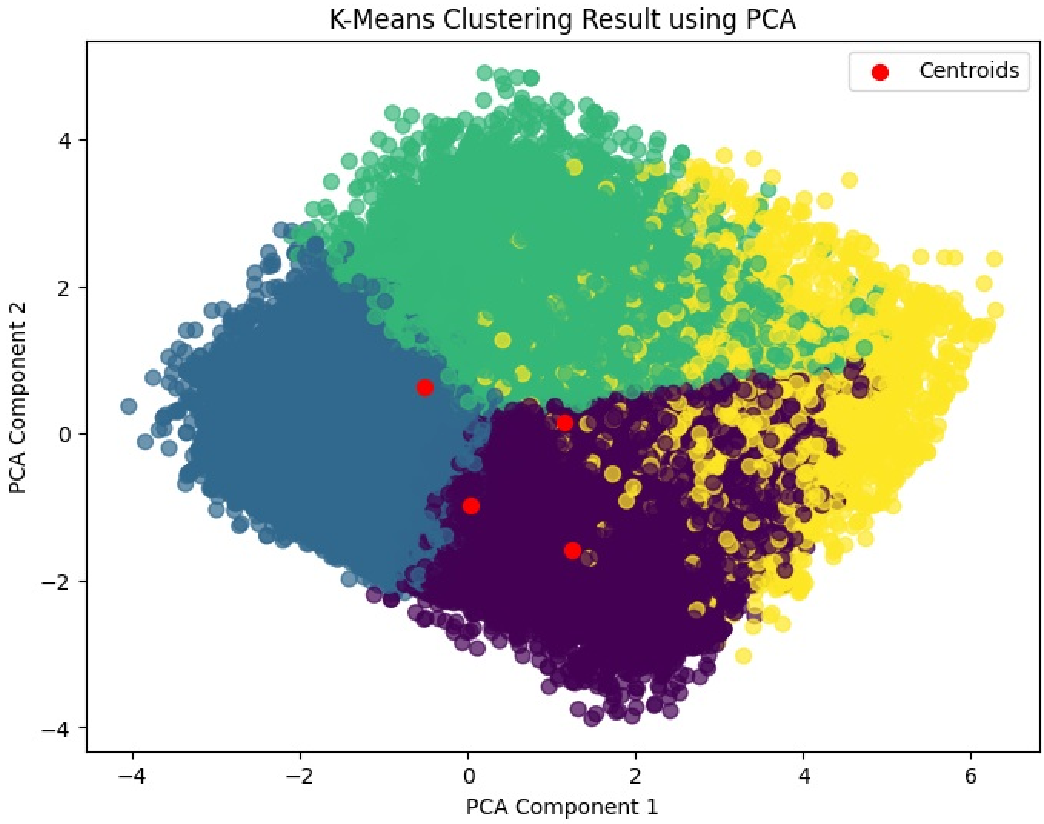
In [59]:
```
from sklearn.metrics import silhouette_score



silhouette_avg = silhouette_score(X_train, kmlabels)
print(f"Silhouette Score: {silhouette_avg}")
```

```
Silhouette Score: 0.28438734858170817
```

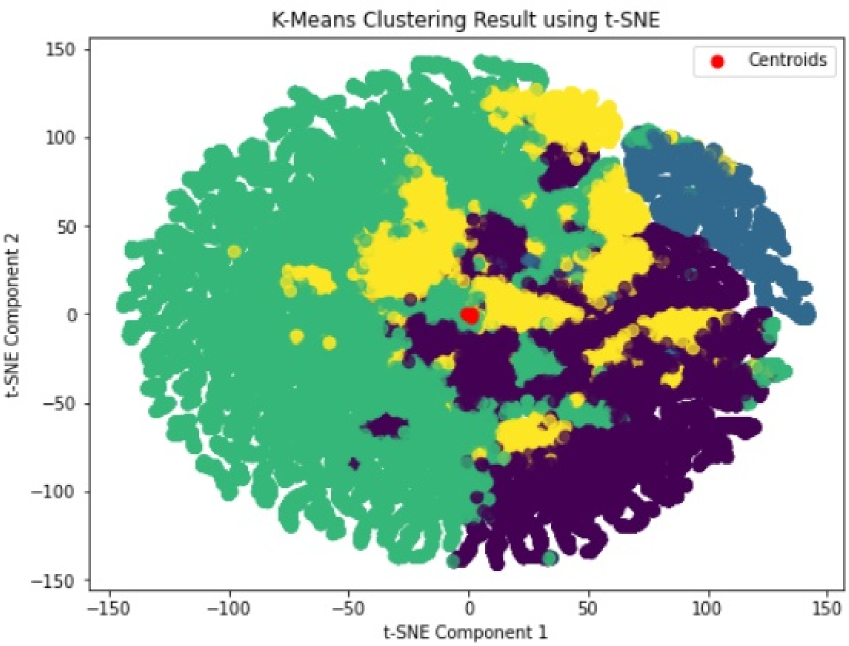
In [60]:
```
overall_mean = np.mean(X_train.values.flatten())
tss = np.sum((X_train.values.flatten() - overall_mean) ** 2)
wcss = kmeans.inertia_
bcss = tss - wcss
print(f"Between-Cluster Sum of Squares (BCSS): {bcss}")
```

```
Between-Cluster Sum of Squares (BCSS): 356842.2091513528
```

In [61]:
```
from sklearn.decomposition import PCA

pca = PCA(2)
X_pca = pca.fit_transform(X_train)

plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=kmlabels, s=50, alpha=0.7)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], c='red', marker='.', s=200, label='Ce
plt.title('K-Means Clustering Result using PCA')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.legend()
plt.show()
```

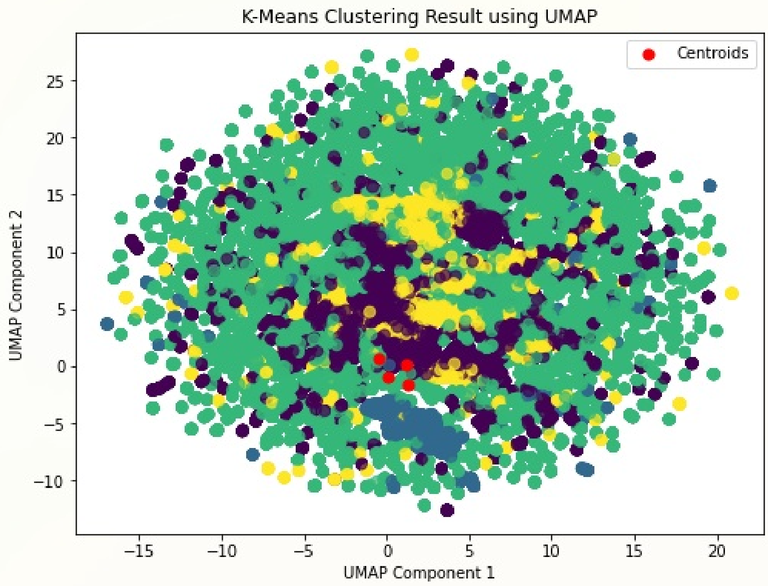


In [62]:
```
# from sklearn.manifold import TSNE


# tsne = TSNE(n_components=2, random_state=42)
# X_tsne = tsne.fit_transform(X_train)


# plt.figure(figsize=(8, 6))
# plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=kmlabels, s=50, alpha=0.7, cmap='viridis')
# plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], c='red', marker='.', s=200, label='(
# plt.title('K-Means Clustering Result using t-SNE')
# plt.xlabel('t-SNE Component 1')
# plt.ylabel('t-SNE Component 2')
# plt.legend()
# plt.show()
```



In [63]:
```
# import umap

# reducer = umap.UMAP(n_components=2)
# X_umap = reducer.fit_transform(X_train)

# plt.figure(figsize=(8, 6))
```

```
# plt.scatter(X_umap[:, 0], X_umap[:, 1], c=kmlabels, s=50, alpha=0.7, cmap='viridis')
# plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], c='red', marker='.', s=200, label='(
# plt.title('K-Means Clustering Result using UMAP')
# plt.xlabel('UMAP Component 1')
# plt.ylabel('UMAP Component 2')
# plt.legend()
# plt.show()
```



In [64]:
```
kmeans.fit(df_agg_processed)

df_agg['Kmeans_clusters']=kmeans.labels_
df_agg.head()
```

Out[64]:

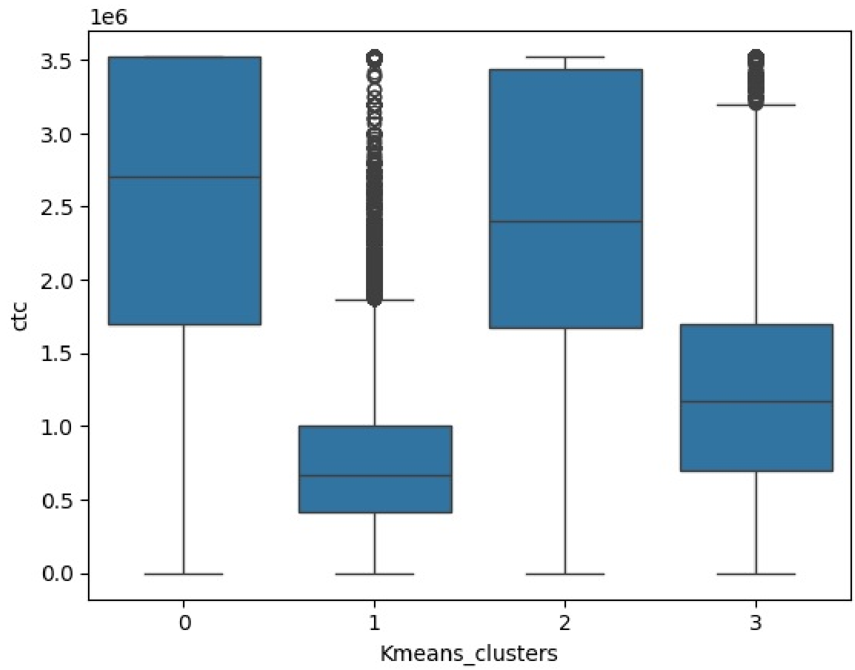
	email_hash	ctc	company_hash	job_position	orgyear	ctc_updated_year	Experi
0	00003288036a44374976948c327f246fdbdf0778546904...	3500000.0	bxwqgogen	backend engineer	2012.0	2019.0	
1	0000aaa0e6b61f7636af1954b43d294484cd151c9b3cf6...	250000.0	nqsn axsxnvr	backend engineer	2013.0	2020.0	
2	0000d58fbc18012bf6fa2605a7b0357d126ee69bc41032...	1300000.0	gunhb	fullstack engineer	2021.0	2019.0	
3	000120d0c8aa304fcf12ab4b85e21feb80a342cfea03d4...	2000000.0	bxwqgotbx wgqugqvnxgz	fullstack engineer	2005.5	2021.0	
4	00014d71a389170e668ba96ae8e1f9d991591acc899025...	3400000.0	fvrbvqn rvmo	NaN	2009.0	2018.0	

In [65]:
```
df_agg['Kmeans_clusters'].value_counts()
```

Out[65]:
```
1    77847
3    41734
2    25494
0     8210
Name: Kmeans_clusters, dtype: int64
```

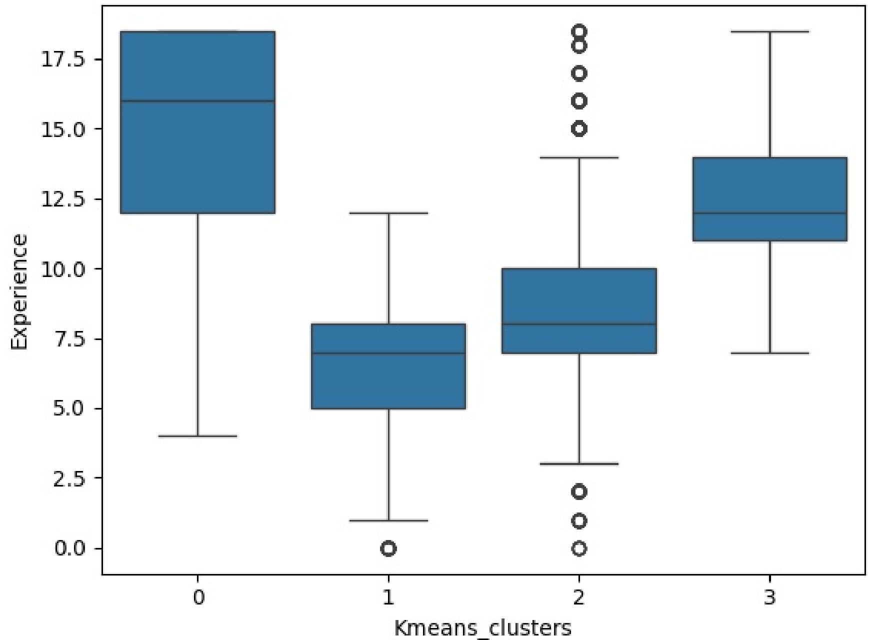
In [66]:
```
sns.boxplot(data=df_agg,x='Kmeans_clusters',y='ctc')
```

Out[66]: <Axes: xlabel='Kmeans_clusters', ylabel='ctc'>



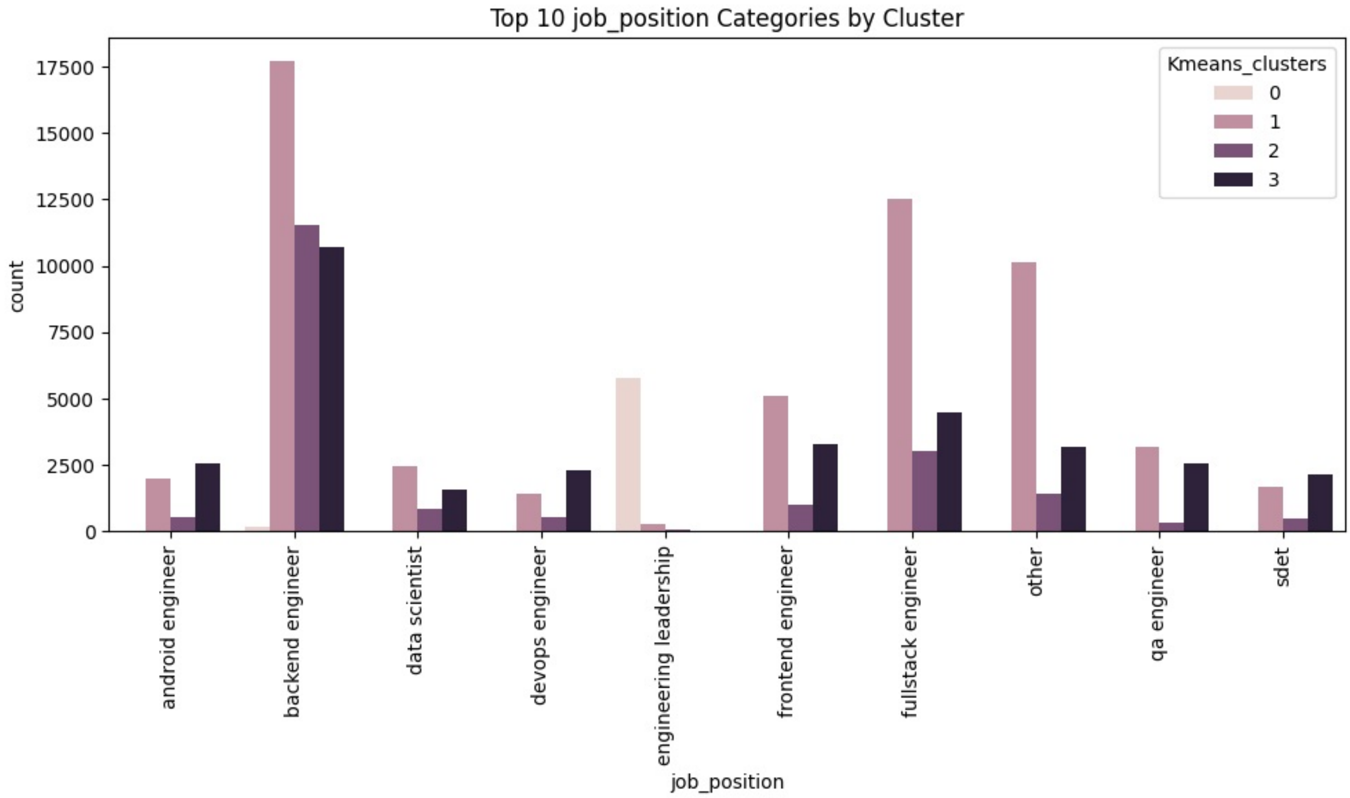
In [67]:
```
sns.boxplot(data=df_agg,x='Kmeans_clusters',y='Experience')
```

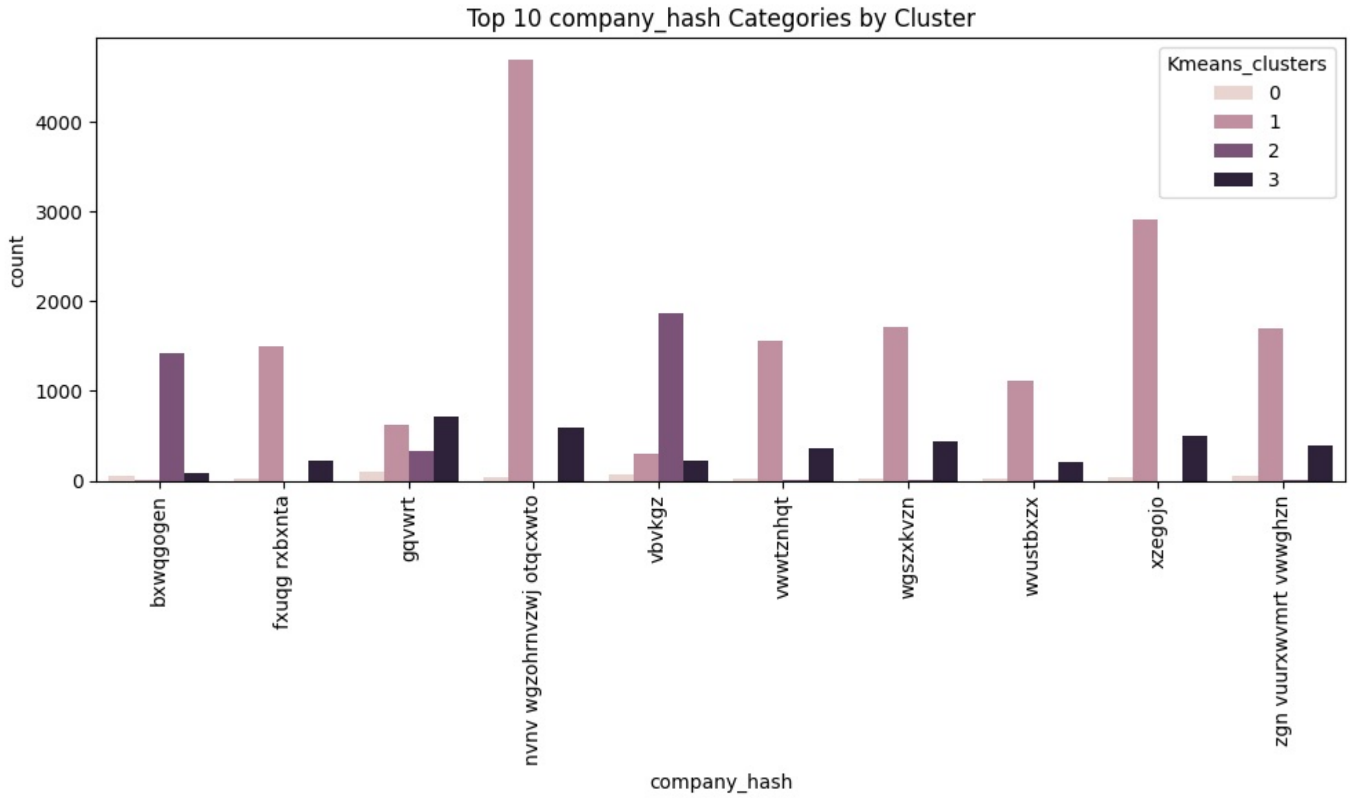
Out[67]: <Axes: xlabel='Kmeans_clusters', ylabel='Experience'>



In [68]:
```
for col in ['job_position', 'company_hash']:

    top_categories = df_agg[col].value_counts().nlargest(10).index
    groupcol = df_agg[df_agg[col].isin(top_categories)].groupby([col, 'Kmeans_clusters']).size().reset_index(nar


    plt.figure(figsize=(10, 6))
    sns.barplot(x=col, y='count', hue='Kmeans_clusters', data=groupcol)
```

```
    plt.title(f'Top 10 {col} Categories by Cluster')
    plt.xticks(rotation=90)
    plt.tight_layout()
    plt.show()
```





- Kmeans could not produce meaningful clusters. Relatively low silhouette score indicates that the clusters are not well-separated. BCSS score is moderately high.

- Clusetr having low ctc and low experience is highest number of values followed by cluster with moderate ctc and higher experience. Cluster having high ctc and Experience has least values.

- In the Cluster having high ctc and moderate experience (7 to 10 years).Most of the employees work as Backend Engineers

- Next Cluster has employees with Low ctc and low experience. Most of them work as Backend and Fullstack engineers. High number of cluster employees work at company 'nvnv wgzohrnvzwj otqcxwto'.

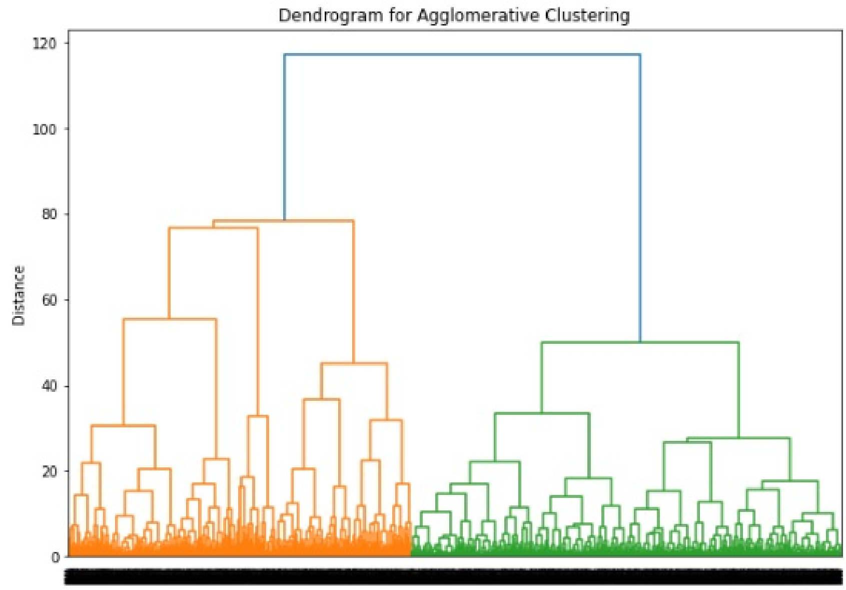
- Cluster with employees have moderate ctc and higher experience mostly work as Backend and full stack engineeers.

- Cluster employees have very high ctc and Experience. They work in mostly Engineering Leadership roles.

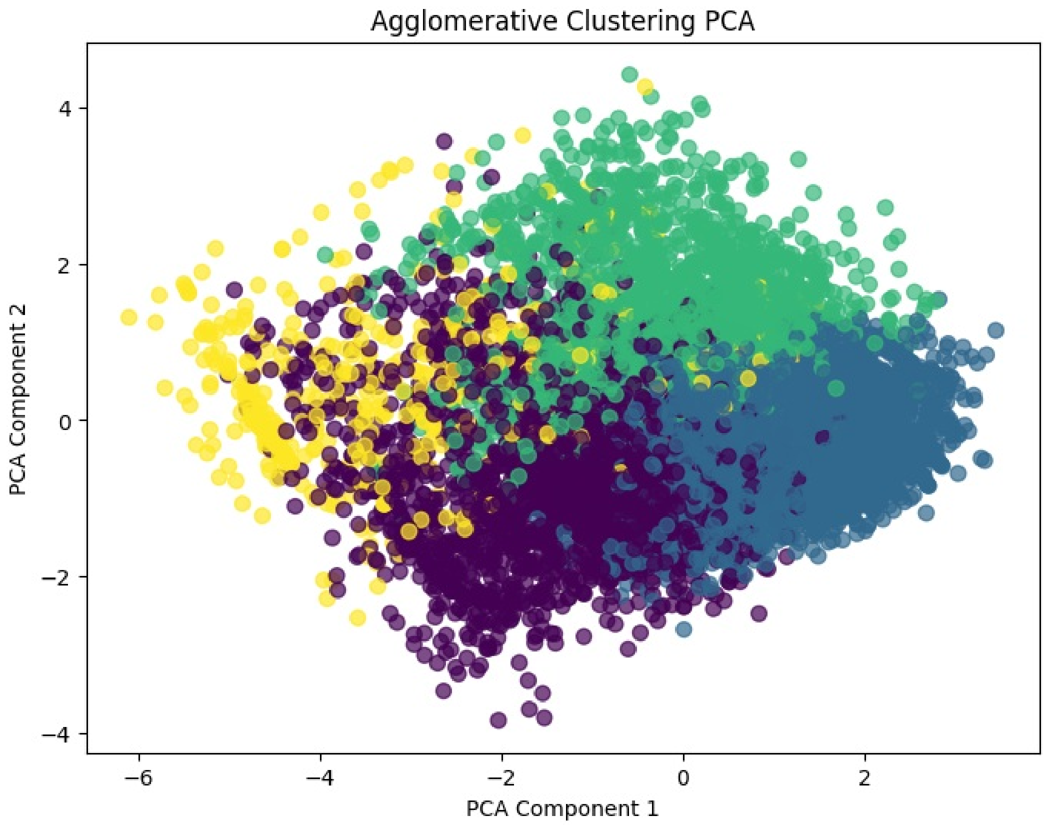
10. Using Agglomerative Clustering.

Visualization using Dendogram to determine number of clusters.

In [69]:
```
# from scipy.cluster.hierarchy import dendrogram, linkage

# np.random.seed(42)
# sample_size = 5000
# random_indices = np.random.choice(X_train.shape[0], size=sample_size, replace=False)
# X_sample = X_train.iloc[random_indices, :]

# linked = linkage(X_sample, method='ward')


# plt.figure(figsize=(10, 7))
# dendrogram(linked,
#            orientation='top',
#            distance_sort='descending',
#            show_leaf_counts=False)
# plt.title('Dendrogram for Agglomerative Clustering')

# plt.ylabel('Distance')
# plt.show()
```



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

np.random.seed(42)


sample_size = 10000
random_indices = np.random.choice(X_train.index, size=sample_size, replace=False)
X_sample = X_train.loc[random_indices, :]


agg_clustering = AgglomerativeClustering(n_clusters=4,  linkage='ward')
agglabels = agg_clustering.fit_predict(X_sample)


pca = PCA(2)
X_pca = pca.fit_transform(X_sample)

plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=agglabels, cmap='viridis', s=50, alpha=0.7)
plt.title('Agglomerative Clustering PCA')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.show()
```

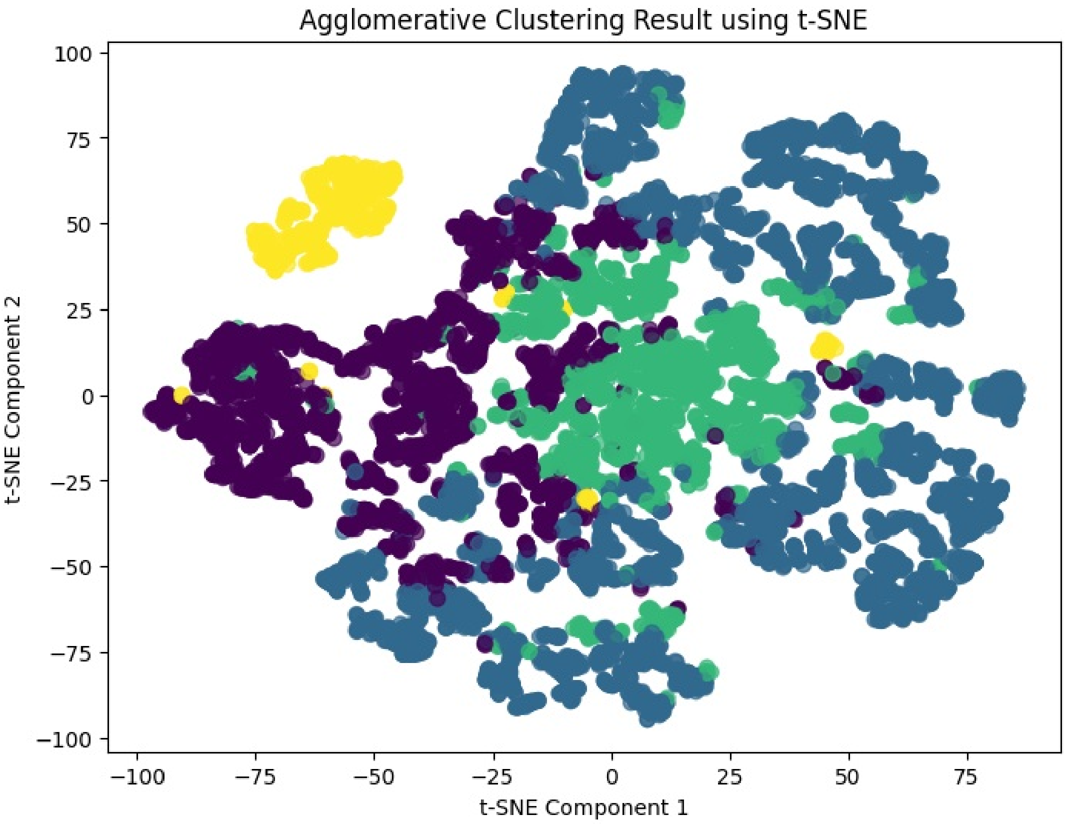


In [71]:
```
silhouette_avg = silhouette_score(X_sample, agglabels)
```

```
print(f"Silhouette Score: {silhouette_avg}")
```

```
Silhouette Score: 0.23645064654367803
```

In [72]:
```
from sklearn.manifold import TSNE


tsne = TSNE(n_components=2, random_state=42)
X_tsne = tsne.fit_transform(X_sample)


plt.figure(figsize=(8, 6))
plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=agglabels, s=50, alpha=0.7, cmap='viridis')

plt.title('Agglomerative Clustering Result using t-SNE')
plt.xlabel('t-SNE Component 1')
plt.ylabel('t-SNE Component 2')

plt.show()
```



In [73]:
```
label_map = pd.Series(agglabels, index=X_sample.index)

df_agg['Agg_Cluster_Labels'] = df_agg.index.map(label_map)

df_agg.head()
```

Out[73]:

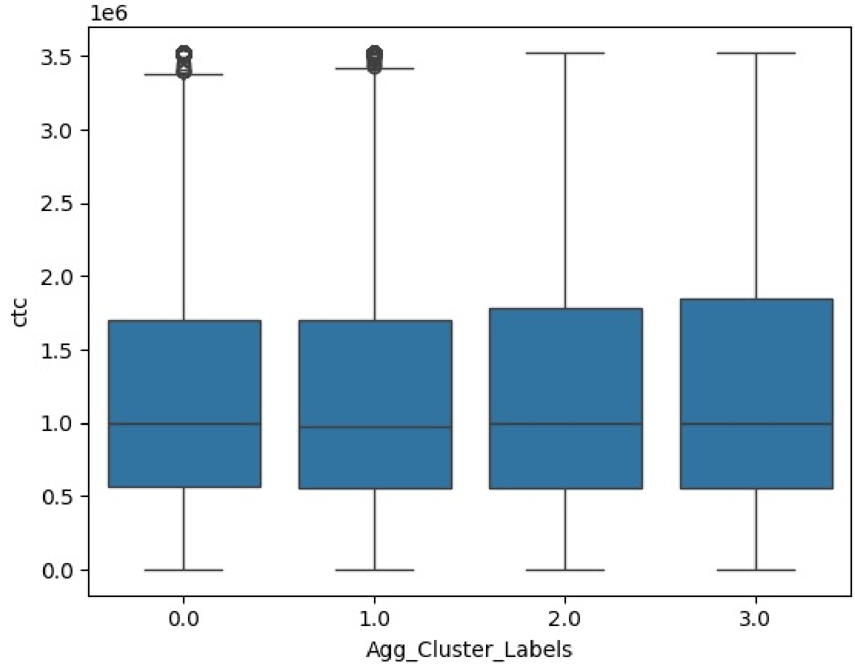
	email_hash	ctc	company_hash	job_position	orgyear	ctc_updated_year	Experi
0	00003288036a44374976948c327f246fdbdf0778546904...	3500000.0	bxwqgogen	backend engineer	2012.0	2019.0	
1	0000aaa0e6b61f7636af1954b43d294484cd151c9b3cf6...	250000.0	nqsn axsxnvr	backend engineer	2013.0	2020.0	
2	0000d58fbc18012bf6fa2605a7b0357d126ee69bc41032...	1300000.0	gunhb	fullstack engineer	2021.0	2019.0	
3	000120d0c8aa304fcf12ab4b85e21feb80a342cfea03d4...	2000000.0	bxwqgotbx wgqugqvnxgz	fullstack engineer	2005.5	2021.0	
4	00014d71a389170e668ba96ae8e1f9d991591acc899025...	3400000.0	fvrbvqn rvmo	NaN	2009.0	2018.0	

In [74]:
```
df_agg['Agg_Cluster_Labels'].value_counts()
```

Out[74]:
```
1.0    4906
0.0    2627
2.0    1920
3.0     539
Name: Agg_Cluster_Labels, dtype: int64
```

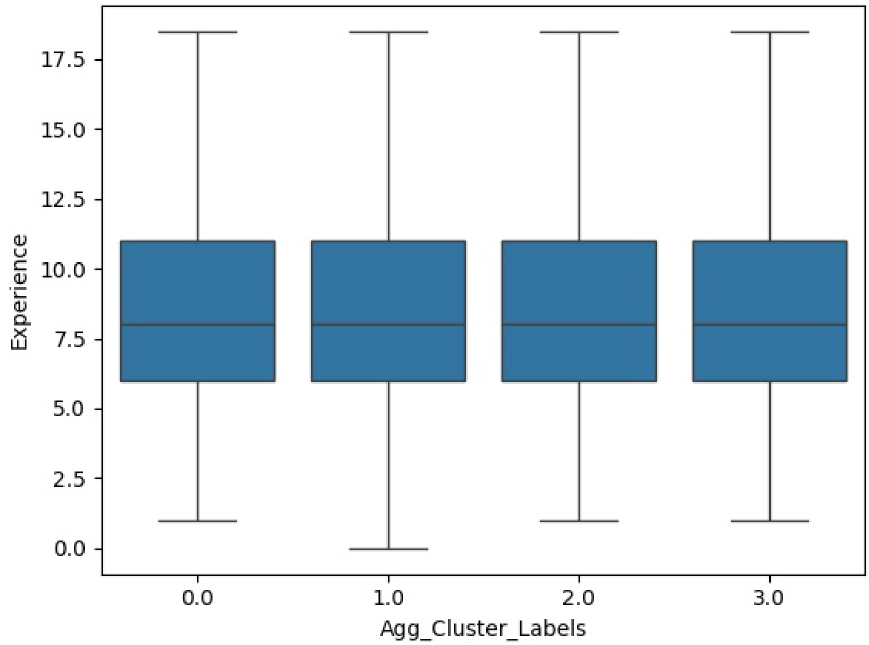
In [75]:
```
sns.boxplot(data=df_agg,x='Agg_Cluster_Labels',y='ctc')
```

Out[75]:
```
<Axes: xlabel='Agg_Cluster_Labels', ylabel='ctc'>
```

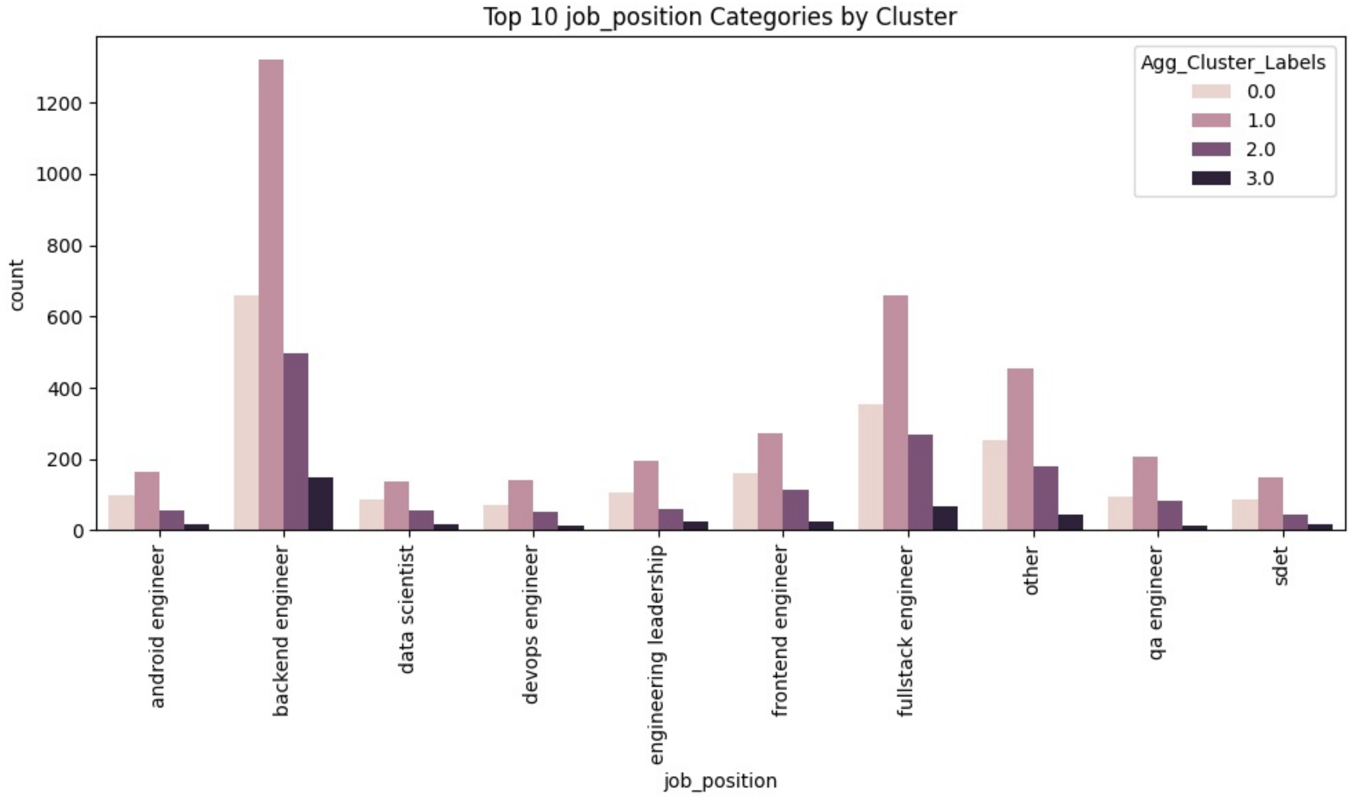


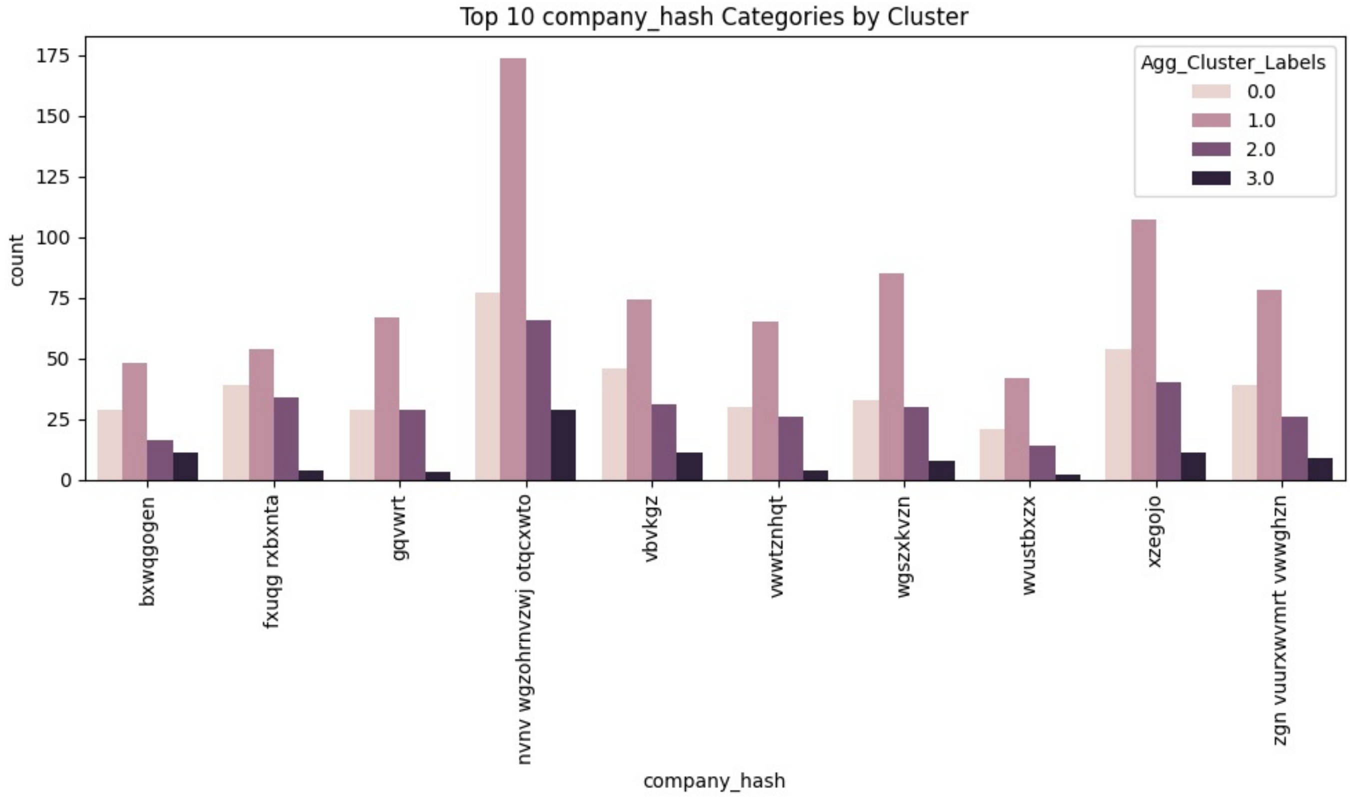
In [76]:
```python
sns.boxplot(data=df_agg,x='Agg_Cluster_Labels',y='Experience')
```

Out[76]:
```
<Axes: xlabel='Agg_Cluster_Labels', ylabel='Experience'>
```



In [77]:
```python
for col in ['job_position', 'company_hash']:

    top_categories = df_agg[col].value_counts().nlargest(10).index
    groupcol = df_agg[df_agg[col].isin(top_categories)].groupby([col, 'Agg_Cluster_Labels']).size().reset_index


    plt.figure(figsize=(10, 6))
    sns.barplot(x=col, y='count', hue='Agg_Cluster_Labels', data=groupcol)



    plt.title(f'Top 10 {col} Categories by Cluster')
    plt.xticks(rotation=90)
    plt.tight_layout()
    plt.show()
```





Agglomerative clustering also has low silhouette score of 0.18 indicates that the clusters are not well-separated. Couldnot see distinction between clusters while analysing using ctc, experience and other columns.

Insights

- The dataset has 153443 unique learners.

- Majority of the learners (22%) are employed at the company 'nvnv wgzohrnvzwj otqcxwto' while 14% work at 'xzegojo' and 10% at 'vbvkgz'.

- 35% of students are having the current job position as Backend Engineering followed by 16% who are Fullstack Engineers.

- The joining year of 13.5% employees at their current company is 2016 closely followed by 2018 an 2017.

- Amost 35% of learners has their CTC updated in the year 2019 and 27% and 24% got their ctc updated in 2021 and 2020 respectively. A total of only 13% students had their ctc updated before the year 2018.

- 50% students are of low income bin, 40% in medium and only 10% learners are of high income bin.

- 74% employees are in job_positions having medium prominance

- It is clear from the plot that the column ctc is extremely right-skewed. This means that the majority of the data points are concentrated on the left which are lower values , but there are a few outliers or extreme values on the right. The column in also having very high value for kurtosis which means it has heavier tails and a sharper peak around the mean compared to a normal distribution.

- The values in the column range from 2 to 1.0e+9. Upon removing the outliers in column ctc, we can see 75% of values lie before 1739999.

- The column orgyear is left skewed with leptokurtic distribution. The values in the column ranges from 1970 to 2024 where 87% of employees joined their current company between 2010 and 2020.

- Experience of employees ranges from 0 to 54 years where the data is rght skewed and leptokurtic. 75% of employees are hving experience of below 11 years.

- Most companies give a ctc of below 2000000 to the learners. Companies like 'bxwqgogen' and 'vbvkgz' are seen to give higher ctc to more employees.

- Higher ctc is given to employees in Engineering Leadership position. All other job positions are given an average ctc of below 3000000.

- Most learners from all companies are showing an experience of below 20 years. Here learners in Engineering Leadership position seems to have more experience compared to others.

- Experience and orgyear are having high negative correlation as experience increases when the learner has joined the company earlier.

- There is a positive correlation between ctc and Experience. With increase in experience, employee can demand more ctc.

- The negative correlation between experience and ctc is because the earlier the employee joins the company, the salary will be more.

The clustering algorithms were not able to produce clearly seperated clusters.

K Means Clustering Outcome

- Kmeans could not produce meaningful clusters. Relatively low silhouette score indicates that the clusters are not well-separated. BCSS score is moderately high.

- Clusetr having low ctc and low experience is highest number of values followed by cluster with moderate ctc and higher experience. Cluster having high ctc and Experience has least values.

- In the Cluster having high ctc and moderate experience (7 to 10 years).Most of the employees work as Backend Engineers

- Next Cluster has employees with Low ctc and low experience. Most of them work as Backend and Fullstack engineers. High number of cluster employees work at company 'nvnv wgzohrnvzwj otqcxwto'.

- Cluster with employees have moderate ctc and higher experience mostly work as Backend and full stack engineeers.
- Cluster employees have very high ctc and Experience. They work in mostly Engineering Leadership roles.

Recommendations

- Students of Cluster having Low CTC and Moderate Experience will be freshers or of young demography.They be can offered specialized training on emerging technologies and domains having future opportunities that will help them to transition their career to fields having higher compensation.
- Cluster members having Moderate CTC and High Experience can be helped to achieve job roles in leadership positions or senior technical roles and train them to gain technical expertise.As they are of older demography aligning them with recent market trends and latest technologies is very important.They also need to be mentored on effectively communicating their experience and skills to negotiate higher salaries
- Segment with High CTC and Moderate Experience can be given programs on Advanced technical skills which can help these learners to take more leadership roles.
- Cluster having very High CTC and high Experience can be provided training on leadership programs, technical decision-making courses and management courses. Mentoring roles could be introduced for this highly experienced group. They need to be taught programs on latest technologies related to the course they have opted for.
- As Companies like 'vbvkgz' and 'bxwqgogen' offer higher CTC to their employees. Scaler can strengthen partnerships with these companies, providing learners with more internship and job placement opportunities.
- With 50% of learners in the low-income bin, Scaler can offer lessons or mentorship on how to negotiate for better salary and provide resources to transition into higher-paying roles. They can also connect learners to companies known for offering higher compensation, like 'bxwqgogen'

and 'vbvkgz'.

- Since a significant portion of learners are Backend Engineers and Fullstack Engineers, Scaler can create highly specialized tracks focused on Backend and Fullstack development. Offering advanced topics and very latest technical knowledge can help these learners further their careers.

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js