# Business Case: Scaler - Clustering

## Problem Statement

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.

## Data Dictionary:

- '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)

In [1]:
```
import pandas as pd
import numpy as np

import re
import datetime

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.decomposition import PCA
from sklearn.manifold import TSNE

from sklearn.preprocessing import StandardScaler, OneHotEncoder
```

```
from sklearn.impute import KNNImputer

# Clsutering Algorithms
from sklearn.cluster import DBSCAN, KMeans, AgglomerativeClustering
import scipy.cluster.hierarchy as sch

from sklearn.metrics import silhouette_score
```

# 1. Exploratory Data Analysis

## 1.1 Reading the dataset and checking its structure

In [2]:
```
data = pd.read_csv("scaler_clustering.csv")
data.head()
```

Out[2]:

| | 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 [3]:
```
data.shape
```

Out[3]: (205843, 7)

In [4]:
```
data.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      153279 non-null  object 
 6   ctc_updated_year  205843 non-null  float64
dtypes: float64(2), int64(2), object(3)
memory usage: 11.0+ MB
```

In [5]:
```
# `Unnamed: 0` column only represent Index of dataset / record number. Dropping the column as it will not impact the clusterin
data.drop(['Unnamed: 0'], axis = 1 , inplace = True)
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205843 entries, 0 to 205842
Data columns (total 6 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   company_hash      205799 non-null  object 
 1   email_hash        205843 non-null  object 
 2   orgyear           205757 non-null  float64
 3   ctc               205843 non-null  int64  
 4   job_position      153279 non-null  object 
 5   ctc_updated_year  205843 non-null  float64
dtypes: float64(2), int64(1), object(3)
memory usage: 9.4+ MB
```

In [7]:
```
data.head()
```

Out[7]:

| | 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 |

## 1.2 Checking and Dropping duplciate records

In [8]:
```
data.duplicated().value_counts()
```

Out[8]:
```
False    205809
True         34
Name: count, dtype: int64
```

In [9]:
```
data.drop_duplicates(inplace = True)
data
```

Out[9]:

| | 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 |
| **...** | ... | ... | ... | ... | ... | ... |
| **205838** | vuurt xzw | 70027b728c8ee901fe979533ed94ffda97be08fc23f33b... | 2008.0 | 220000 | NaN | 2019.0 |
| **205839** | husqvawgb | 7f7292ffad724ebbe9ca860f515245368d714c84705b42... | 2017.0 | 500000 | NaN | 2020.0 |
| **205840** | vwwgrxnt | cb25cc7304e9a24facda7f5567c7922ffc48e3d5d6018c... | 2021.0 | 700000 | NaN | 2021.0 |
| **205841** | zgn vuurxwvmrt | fb46a1a2752f5f652ce634f6178d0578ef6995ee59f6c8... | 2019.0 | 5100000 | NaN | 2019.0 |
| **205842** | bgqsvz onvzrtj | 0bcfc1d05f2e8dc4147743a1313aa70a119b41b30d4a1f... | 2014.0 | 1240000 | NaN | 2016.0 |

205809 rows × 6 columns

In [10]:
```
data.duplicated().value_counts()
```

Out[10]:
```
False    205809
Name: count, dtype: int64
```

## 1.3 Missing value check

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

Out[11]:
```
company_hash          44
email_hash             0
orgyear               86
ctc                    0
job_position       52548
ctc_updated_year       0
dtype: int64
```

In [12]:
```
# Percentage of null
(data.isna().sum()/data.shape[0])*100
```

Out[12]:
```
company_hash         0.021379
email_hash           0.000000
orgyear              0.041786
ctc                  0.000000
job_position        25.532411
ctc_updated_year     0.000000
dtype: float64
```

## 1.4 Statistical Summary of Dataset

In [13]:
```
data.describe()
```

Out[13]:

| | orgyear | ctc | ctc_updated_year |
|---|---|---|---|
| **count** | 205723.000000 | 2.058090e+05 | 205809.000000 |
| **mean** | 2014.882264 | 2.271862e+06 | 2019.628272 |
| **std** | 63.576352 | 1.180187e+07 | 1.325187 |
| **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 [14]:
```
data.describe(include = object)
```

Out[14]:

| | company_hash | email_hash | job_position |
|---|---|---|---|
| **count** | 205765 | 205809 | 153261 |
| **unique** | 37299 | 153443 | 1016 |
| **top** | nvnv wgzohrnvzwj otqcxwto | bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7... | Backend Engineer |
| **freq** | 8337 | 10 | 43546 |

## 1.5 Imputation and Cleaning the text (object) column data

- Using Mean Imputation for numeric column and Arbitrary imputation for Object Column
- Removing whitespaces at end and start
- Remove Special characters from the dataset by using Regex

In [15]:
```
# Removing any whitespaces at end or start of text if any
data['company_hash'] = data['company_hash'].str.strip()
data['email_hash'] = data['email_hash'].str.strip()

data['job_position'] = data['job_position'].str.strip().str.lower()
data['orgyear'] = data['orgyear'].fillna(data['orgyear'].mean())
```

In [16]:
```
# Function to clean special characters using Regex
def clean_text(text):
    return re.sub('[^A-Za-z0-9 ]+', '', text)

# Apply the function to both columns
data['company_hash'] = data['company_hash'].fillna('').apply(clean_text)
data['email_hash']  = data['email_hash'].fillna('').apply(clean_text)
data['job_position'] = data['job_position'].fillna('').apply(clean_text)
```

## 1.5 Unique Values across each column

In [17]:
```
obj_col = ['company_hash', 'email_hash', 'job_position']
year_col = ['orgyear', 'ctc_updated_year']
int_col = ['ctc']
```

## Unique values in company_hash, email_hash and job_position

In [18]:
```
for i in obj_col:
    print(f" \nUnique value of Column = {i} : ")
    print(data[i].value_counts(dropna =False))
```

```
Unique value of Column = company_hash :
company_hash
nvnv wgzohrnvzwj otqcxwto         8337
xzegojo                           5381
vbvkgz                            3481
zgn vuurxwvmrt vwwghzn            3410
wgszxkvzn                         3240
                                  ...
onvqmhwpo                            1
bvsxw ogenfvqt uqxcvnt rxbxnta       1
agsbv ojontbo                        1
vnnhzt xzegwgb                       1
bvptbjnqxu td vbvkgz                 1
Name: count, Length: 37300, dtype: int64

Unique value of Column = email_hash :
email_hash
bbace3cc586400bbc65765bc6a16b77d8913836cfc98b77c05488f02f5714a4b    10
3e5e49daa5527a6d5a33599b238bf9bf31e85b9efa9a94f1c88c5e15a6f31378     9
6842660273f70e9aa239026ba33bfe82275d6ab0d20124021b952b5bc3d07e6c     9
298528ce3160cc761e4dc37a07337ee2e0589df251d73645aae209b010210eee     9
c0eb129061675da412b0deb15871dd06ef0d7cd86eb5f7e8cc6a20b0d1938183     8
                                                                    ..
63933d31becd1487d93d56844919896334e3ae39c4095979816c6fbb8816153a     1
23bcc14067e0fec60b8772b3e20abbb8fa9f2146738d37056e0d20d33a97c690     1
5a1c9d9a745d6ee95136047698dba8f68f00bac522de6d83d18cf062f7286e22     1
062597458dc597d35b2dbf3e417ac160244dc8c3dd50fce716837dc1e6fc7a10     1
0bcfc1d05f2e8dc4147743a1313aa70a119b41b30d4a1f7e738a6a87d3712c31     1
Name: count, Length: 153443, dtype: int64

Unique value of Column = job_position :
job_position
                                    52550
backend engineer                    43546
fullstack engineer                  25976
other                               18072
frontend engineer                   10417
                                    ...
area operations manager                 1
risk investigator                       1
```

```
x                                    1
machine learning engineer intern     1
azure data factory                   1
Name: count, Length: 899, dtype: int64
```

## Seeing Different job positions ( as we will apply one-hot encoding on it)

In [19]:
```
data['job_position'].str.contains('intern').value_counts()
```

Out[19]:
```
job_position
False    203028
True       2781
Name: count, dtype: int64
```

In [20]:
```
data['job_position'].str.contains('student').value_counts()
```

Out[20]:
```
job_position
False    205779
True         30
Name: count, dtype: int64
```

In [21]:
```
data['job_position'].str.contains('engineer').value_counts()
```

Out[21]:
```
job_position
True     114553
False     91256
Name: count, dtype: int64
```

In [22]:
```
data['job_position'].str.contains('analyst').value_counts()
```

Out[22]:
```
job_position
False    202784
True       3025
Name: count, dtype: int64
```

In [23]:
```
data['job_position'].str.contains('scientist').value_counts()
```

Out[23]:
```
job_position
False    200426
True       5383
Name: count, dtype: int64
```

In [24]:
```
data['job_position'].str.contains('designer').value_counts()
```

Out[24]:
```
job_position
False    204490
True       1319
Name: count, dtype: int64
```

In [25]:
```
data['job_position'].str.contains('developer').value_counts()
```

Out[25]:
```
job_position
False    205635
True        174
Name: count, dtype: int64
```

In [26]:
```
# Top 10 job positions
(data['job_position'].value_counts()[:10] / data.shape[0] ) *100
```

Out[26]:
```
job_position
                          25.533383
backend engineer          21.158453
fullstack engineer        12.621411
other                      8.780957
frontend engineer          5.061489
engineering leadership     3.338046
qa engineer                3.200540
data scientist             2.607758
android engineer           2.602413
sdet                       2.413403
Name: count, dtype: float64
```

In [27]:
```
sum((data['job_position'].value_counts()[:10] / data.shape[0] ) *100)
```

Out[27]:
```
87.31785296075488
```

### Unique values in orgyear and ctc_updated_year

In [28]:
```
# Correcting the data type of year column to int (from flaot). Cannot convert orgyear to int now as it has missing values in i
data['ctc_updated_year'] = data['ctc_updated_year'].astype('int')
```

In [29]:
```
# Unique values in orgyear and ctc_updated_year
for i in year_col:
    print(f" \nUnique value of Column = {i} : ")
    print(data[i].value_counts(dropna =False))
```

```
Unique value of Column = orgyear :
orgyear
2018.0    25253
2019.0    23420
2017.0    23233
2016.0    23038
2015.0    20609
          ...
2107.0        1
1972.0        1
2101.0        1
208.0         1
200.0         1
Name: count, Length: 78, dtype: int64

Unique value of Column = ctc_updated_year :
ctc_updated_year
2019    68665
2021    64974
2020    49435
2017     7561
2018     6746
2016     5501
2015     2927
Name: count, dtype: int64
```

### Checking for invalid records in orgyear beside the NaN values.

- Considering 2025 as invalid record
- Considering 2025 as valid record ( as maybe we have records of people starting job from next year as we have thier data now in 2024)

In [30]:
```python
# Checking for invalid records in orgyear beside the NaN values. ( Consiering 2025 as invalid record)
data.loc[ ( (data['orgyear']> 2024) | (data['orgyear'] < 1000) ),: ]
```

Out[30]:

| | company_hash | email_hash | orgyear | ctc | job_position | ctc_updated_year |
|---|---|---|---|---|---|---|
| **2211** | phrxkv | 3394674bb6bb1de6289e931853fa0bd131c811e0054a92... | 2031.0 | 1500000 | backend engineer | 2020 |
| **3651** | wgszxkvzn | 2cc6bae4e52677d27ce3fca38d7a01ecbe537e1dc1c48d... | 2106.0 | 600000 | | 2021 |
| **10076** | xzegojo | 4c171381270155fb87b885f89cd71ca37ebbb8fd9da58b... | 2025.0 | 360000 | other | 2020 |
| **11081** | exqon vacvznvst uqxcvnt rxbxnta | d6df76c2b61fa3a068e4e3812be12a58f86f78a31fe888... | 2029.0 | 310000 | other | 2020 |
| **13424** | 9xntwyzgrgsj | 854ff163ded87211b944dfcaebdcf9e8efa45defc9582f... | 0.0 | 700000 | | 2021 |
| **...** | ... | ... | ... | ... | ... | ... |
| **193131** | vxqvoxv | 0a5e691a0f8c2c06862ef19d43dc11c22f462f800db26b... | 0.0 | 800000 | | 2019 |
| **196354** | vaxnjv mxqrv wvuxnvr | 069308440811d578c817c05392f97e8919baac6aa12aa3... | 1.0 | 2900000 | data scientist | 2019 |
| **198187** | xb v onhatzn | 9429a19771ae913f169917d380c94f003115aaaf904388... | 2025.0 | 300000 | other | 2021 |
| **202210** | mqvmtzatq | d66f939c4318c1958be5bc9e7b70b741aa61be7493ff58... | 2028.0 | 1300000 | backend engineer | 2021 |
| **203992** | xatv ouvqp ogrhnxgzo ucn rna | 7191da2e57dcb0c1301711e889ea72d5cc801e039359b1... | 20165.0 | 850000 | | 2019 |

86 rows × 6 columns

In [31]:
```python
# Checking for invalid records in orgyear beside the NaN values. ( Consiering 2025 as valid record)
data.loc[ ( (data['orgyear']> 2025) | (data['orgyear'] < 1000) ), : ]
```

Out[31]:

| | company_hash | email_hash | orgyear | ctc | job_position | ctc_updated_year |
|---|---|---|---|---|---|---|
| **2211** | phrxkv | 3394674bb6bb1de6289e931853fa0bd131c811e0054a92... | 2031.0 | 1500000 | backend engineer | 2020 |
| **3651** | wgszxkvzn | 2cc6bae4e52677d27ce3fca38d7a01ecbe537e1dc1c48d... | 2106.0 | 600000 | | 2021 |
| **11081** | exqon vacvznvst uqxcvnt rxbxnta | d6df76c2b61fa3a068e4e3812be12a58f86f78a31fe888... | 2029.0 | 310000 | other | 2020 |
| **13424** | 9xntwyzgrgsj | 854ff163ded87211b944dfcaebdcf9e8efa45defc9582f... | 0.0 | 700000 | | 2021 |
| **13698** | oxtbtzo | 4a64fdec422e657b175d5dd914b91e0df7c78ec7716bfe... | 208.0 | 500000 | | 2020 |
| **...** | ... | ... | ... | ... | ... | ... |
| **188672** | wxowg cxatg ntwyzgrgsxto xzaxv ucn rna | c3cce99fc54361b5c213f8043505d2990c8dfa93669df8... | 200.0 | 3000000 | engineering leadership | 2019 |
| **193131** | vxqvoxv | 0a5e691a0f8c2c06862ef19d43dc11c22f462f800db26b... | 0.0 | 800000 | | 2019 |
| **196354** | vaxnjv mxqrv wvuxnvr | 069308440811d578c817c05392f97e8919baac6aa12aa3... | 1.0 | 2900000 | data scientist | 2019 |
| **202210** | mqvmtzatq | d66f939c4318c1958be5bc9e7b70b741aa61be7493ff58... | 2028.0 | 1300000 | backend engineer | 2021 |
| **203992** | xatv ouvqp ogrhnxgzo ucn rna | 7191da2e57dcb0c1301711e889ea72d5cc801e039359b1... | 20165.0 | 850000 | | 2019 |

73 rows × 6 columns

## Unique values in ctc column

In [32]:

```
# Unique values in ctc column
for i in int_col:
    print(f" \nUnique value of Column = {i} : ")
    print(data[i].value_counts(dropna =False))
```

```
Unique value of Column = ctc :
ctc
600000     7831
400000     7598
1000000    7578
500000     7241
800000     6750
           ...
5340000       1
2305000       1
4225000       1
989999        1
3327000       1
Name: count, Length: 3360, dtype: int64
```

In [33]:
```
data['ctc'].max()
```

Out[33]:
```
1000150000
```

In [34]:
```
# Binning ctc column and creating a new column (just for analysis and plots)
bins = [0, 100000, 500000, 1000000, 1500000, 10000000, 15000000,data['ctc'].max()]
labels = ['0-100K', '100K-500K', '500K-1000K', '1000K-1500K', '1500K-10000K', '10000K-15000K', 'More than 15000K']
data['ctc_binned'] = pd.cut(data['ctc'], bins=bins, labels=labels)
```

In [35]:
```
(data['ctc_binned'].value_counts() / data.shape[0] )*100
```

Out[35]:
```
ctc_binned
500K-1000K          30.323261
1500K-10000K        27.272374
100K-500K           21.544733
1000K-1500K         16.902565
0-100K               2.832724
More than 15000K     0.929988
10000K-15000K        0.194355
Name: count, dtype: float64
```

## 1.6 Outlier Check

In [36]:
```python
def outlier_count(data_check):
    data_ = data_check.describe()
    data_.loc['IQR',:] = data_.loc['75%',:] - data_.loc['25%',:]
    data_.loc['UW',:] = data_.loc['75%',:] + ( 1.5 * data_.loc['IQR',:] )
    data_.loc['LW',:] = data_.loc['25%',:] - ( 1.5 * data_.loc['IQR',:] )

    for i in data_.columns:
        data_.loc['Outlier_Count',i] = data_check[(data_check[i] > data_.loc['UW',i]) | (data_check[i] < data_.loc['LW',i])].s

    return data_
```

In [37]:
```python
outlier_count(data)
```

Out[37]:

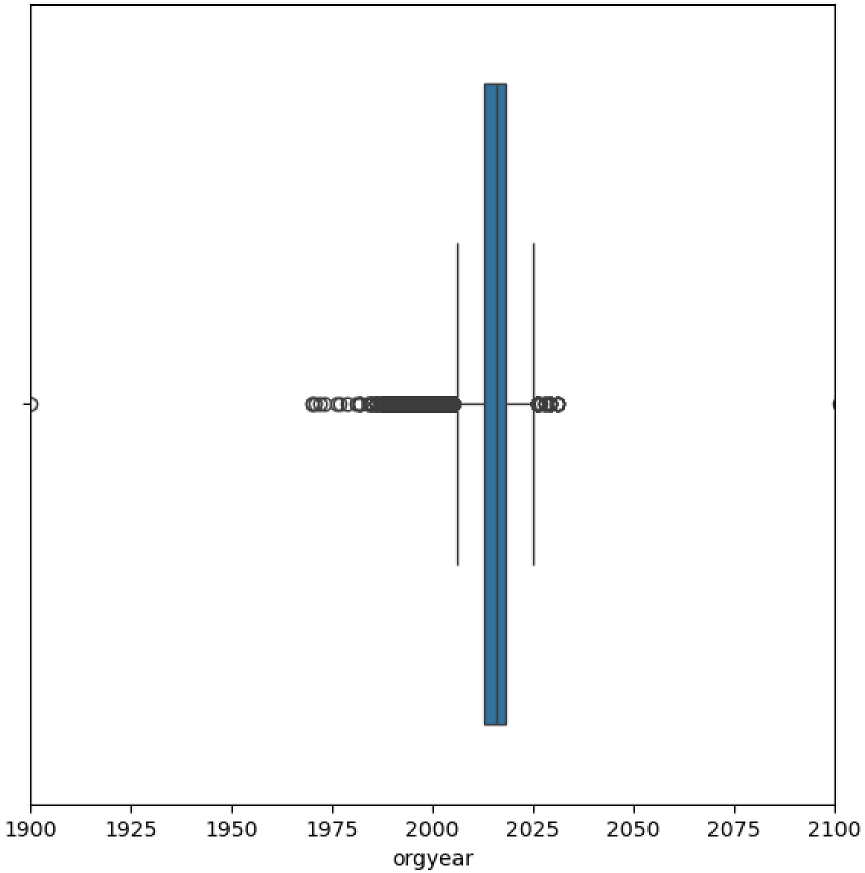
| | orgyear | ctc | ctc_updated_year |
|---|---|---|---|
| **count** | 205809.000000 | 2.058090e+05 | 205809.000000 |
| **mean** | 2014.882264 | 2.271862e+06 | 2019.628272 |
| **std** | 63.563068 | 1.180187e+07 | 1.325187 |
| **min** | 0.000000 | 2.000000e+00 | 2015.000000 |
| **25%** | 2013.000000 | 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 |
| **IQR** | 5.000000 | 1.170000e+06 | 2.000000 |
| **UW** | 2025.500000 | 3.455000e+06 | 2024.000000 |
| **LW** | 2005.500000 | -1.225000e+06 | 2016.000000 |
| **Outlier_Count** | 7764.000000 | 1.312600e+04 | 2927.000000 |

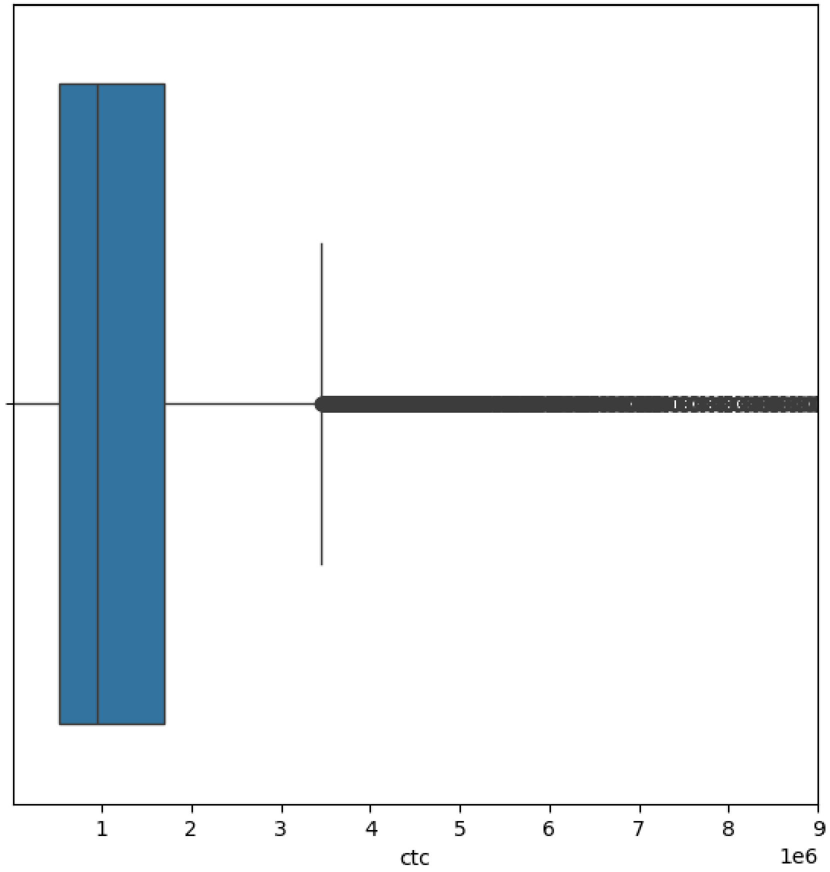
In [38]:
```python
# Checking maximum outlier count percentage compared to total records
```

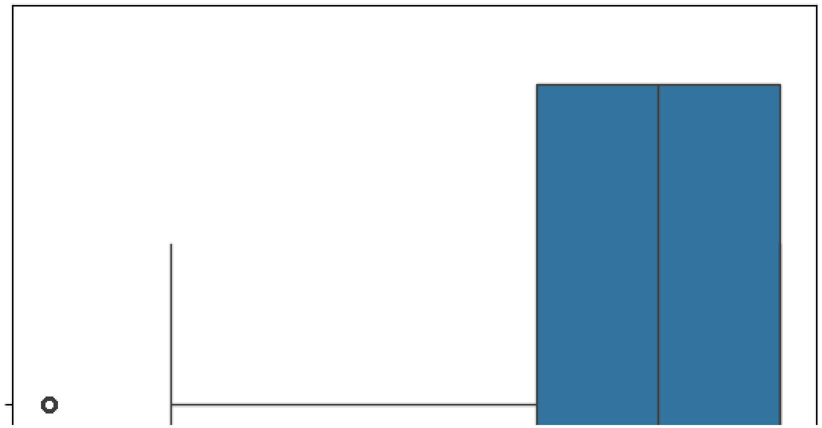
```
(outlier_count(data).loc['Outlier_Count',:] / data.shape[0]) *100
```

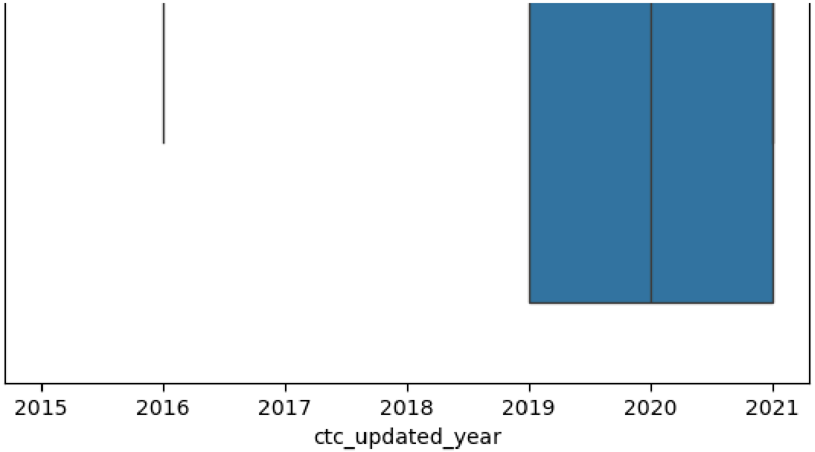
Out[38]:
```
orgyear              3.772430
ctc                  6.377758
ctc_updated_year     1.422192
Name: Outlier_Count, dtype: float64
```

In [39]:
```
plt.figure(figsize=(15, 15))

plt.subplot(221)
box1 = sns.boxplot( x = data['orgyear'])
box1.set_xlim(1900, 2100)

plt.subplot(222)
box2 = sns.boxplot( x = data['ctc'])
box2.set_xlim(10000, 9000000)


plt.subplot(223)
sns.boxplot( x = data['ctc_updated_year'])

plt.show()
```


1900
1925
1950
1975
2000
2025
2050
2075
2100
orgyear


1
2
3
4
5
6
7
8
9
1e6
ctc






In [ ]:

## Observation:

- There are 205843 records and 7 columns in dataset
- `Unnamed: 0` column only represent Index of dataset / record number. Dropping the column as it will not impact the clustering.
- 34 duplicated records found , dropped them. Now the shape of dataset is 205809 rows × 6 columns
- Columns `company_hash` , `orgyear` and `job_position` have null values present in it. Around 25.5% of records in `job_position` are nulls/missing.
- Converted data type of `ctc_updated_year` to int as it represent Year and has all valid values.
- `orgyear` has 86 (cosidering 2025 orgyear also as invlaid) invalid records OR 73 (cosidering 2025 orgyear as vlaid) invalid records in it (besides NaN values). It has invalid values like like 2031 , 2106 , 0 , 20165 , etc. which is not possible considering it Employment start date (year) .
- The `email_hash` - bbace3cc586400bbc65765bc6a16b77d8913836cfc98b77c05488f02f5714a4b is most occuring out of all other. It appears 10 times which means
- Top 10 valid (non nan) job positions based on record count are 'backend engineer', 'fullstack engineer', 'other', 'frontend engineer', 'engineering leadership', 'qa engineer', 'data scientist' 'android engineer', 'sde and , 'devops enginee. Together top 10 job positions make up 64% of dataset.
- Count of outliers is less in all 3 numeric columns. Outlier percentage in `ctc` column is 6.3% , `orgyear` is 3.7% and in `ctc_updated_year` is 1.4%.

- Binned ctc column and created a new column `ctc_binned` and found that around 30% of records have current ctc in range 500K-1000K , 27% of records have ctc in range 1500K-10000K and 22% of records have ctc in range 100K-500K. Very few(around 1%) records have ctc in range 10000K-15000K or more.r'

# 2. Fature Engineering

In [40]:
```
data.isna().sum()
```

Out[40]:
```
company_hash        0
email_hash          0
orgyear             0
ctc                 0
job_position        0
ctc_updated_year    0
ctc_binned          0
dtype: int64
```

## 2.1 Feature Creation

### Created new numeric datatype columns by encoding object columns. We can then use that in clustering (instead of their object column counterpart)

- Grouping by company_hash and calculating the median Cost to Company (CTC) of its employees can be a useful encoding approach for clustering.
- Group by email_hash and calculating the average CTC for each unique email_hash can be another a useful encoding approach for clustering.
- Group by job_position and calculating the average CTC for each unique job_position can be another a useful encoding approach for clustering.
- Group by company_hash and job_position and calculating the average CTC for each unique company_hash and job_position combination can be another a useful encoding approach for clustering.
- Group by email_hash and finding number of unique job_position can be another a useful encoding approach for clustering.
- Group by company_hash and email_hash then finding number of unique orgyear can we a way to find how many times a person salary got incremented/ they got promoted in same company.

In [41]:
```
# Calculate average CTC by company_hash
company_med_ctc = data.groupby('company_hash')['ctc'].median().reset_index()
company_med_ctc.rename(columns={'ctc': 'company_median_CTC'}, inplace=True)

# Merge average CTC back to original dataframe
data = data.merge(company_med_ctc, on='company_hash' , how ="left")
data.head()
```

Out[41]:

| | company_hash | email_hash | orgyear | ctc | job_position | ctc_updated_year | ctc_binned | com |
|---|---|---|---|---|---|---|---|---|
| 0 | atrgxnnt xzaxv | 6de0a4417d18ab14334c3f43397fc13b30c35149d70c05... | 2016.0 | 1100000 | other | 2020 | 1000K-1500K | |
| 1 | qtrxvzwt xzegwgbb rxbxnta | b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10... | 2018.0 | 449999 | fullstack engineer | 2019 | 100K-500K | |
| 2 | ojzwnvwnxw vx | 4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9... | 2015.0 | 2000000 | backend engineer | 2020 | 1500K-10000K | |
| 3 | ngpgutaxv | effdede7a2e7c2af664c8a31d9346385016128d66bbc58... | 2017.0 | 700000 | backend engineer | 2019 | 500K-1000K | |
| 4 | qxen sqghu | 6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520... | 2017.0 | 1400000 | fullstack engineer | 2019 | 1000K-1500K | |

In [42]:
```
email_avg_ctc = data.groupby('email_hash')['ctc'].mean().reset_index()
email_avg_ctc.rename(columns={'ctc': 'email_avg_CTC'}, inplace=True)

data = data.merge(email_avg_ctc, on='email_hash', how = "left")
data.head()
```

Out[42]:

| | company_hash | email_hash | orgyear | ctc | job_position | ctc_updated_year | ctc_binned | comp |
|---|---|---|---|---|---|---|---|---|
| 0 | atrgxnnt xzaxv | 6de0a4417d18ab14334c3f43397fc13b30c35149d70c05... | 2016.0 | 1100000 | other | 2020 | 1000K-1500K | |
| 1 | qtrxvzwt xzegwgbb rxbxnta | b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10... | 2018.0 | 449999 | fullstack engineer | 2019 | 100K-500K | |
| 2 | ojzwnvwnxw vx | 4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9... | 2015.0 | 2000000 | backend engineer | 2020 | 1500K-10000K | |
| 3 | ngpgutaxv | effdede7a2e7c2af664c8a31d9346385016128d66bbc58... | 2017.0 | 700000 | backend engineer | 2019 | 500K-1000K | |
| 4 | qxen sqghu | 6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520... | 2017.0 | 1400000 | fullstack engineer | 2019 | 1000K-1500K | |

In [43]:

```
job_pos_avg_ctc = data.groupby('job_position')['ctc'].mean().reset_index()
job_pos_avg_ctc.rename(columns={'ctc': 'job_pos_avg_CTC_accross_all_companies'}, inplace=True)

data = data.merge(job_pos_avg_ctc, on='job_position', how = "left")
data.head()
```

Out[43]:

| | company_hash | email_hash | orgyear | ctc | job_position | ctc_updated_year | ctc_binned | comp |
|---|---|---|---|---|---|---|---|---|
| **0** | atrgxnnt xzaxv | 6de0a4417d18ab14334c3f43397fc13b30c35149d70c05... | 2016.0 | 1100000 | other | 2020 | 1000K-1500K | |
| **1** | qtrxvzwt xzegwgbb rxbxnta | b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10... | 2018.0 | 449999 | fullstack engineer | 2019 | 100K-500K | |
| **2** | ojzwnvwnxw vx | 4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9... | 2015.0 | 2000000 | backend engineer | 2020 | 1500K-10000K | |
| **3** | ngpgutaxv | effdede7a2e7c2af664c8a31d9346385016128d66bbc58... | 2017.0 | 700000 | backend engineer | 2019 | 500K-1000K | |
| **4** | qxen sqghu | 6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520... | 2017.0 | 1400000 | fullstack engineer | 2019 | 1000K-1500K | |

In [44]:

```
job_pos_avg_ctc_company = data.groupby(['company_hash','job_position'])['ctc'].mean().reset_index()
job_pos_avg_ctc_company.rename(columns={'ctc': 'job_pos_avg_CTC_accross_same_company'}, inplace=True)

data = data.merge(job_pos_avg_ctc_company, on=['company_hash','job_position'], how = "left")
data.head()
```

Out[44]:

| | company_hash | email_hash | orgyear | ctc | job_position | ctc_updated_year | ctc_binned | com |
|---|---|---|---|---|---|---|---|---|
| **0** | atrgxnnt xzaxv | 6de0a4417d18ab14334c3f43397fc13b30c35149d70c05... | 2016.0 | 1100000 | other | 2020 | 1000K-1500K | |
| **1** | qtrxvzwt xzegwgbb rxbxnta | b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10... | 2018.0 | 449999 | fullstack engineer | 2019 | 100K-500K | |
| **2** | ojzwnvwnxw vx | 4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9... | 2015.0 | 2000000 | backend engineer | 2020 | 1500K-10000K | |
| **3** | ngpgutaxv | effdede7a2e7c2af664c8a31d9346385016128d66bbc58... | 2017.0 | 700000 | backend engineer | 2019 | 500K-1000K | |
| **4** | qxen sqghu | 6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520... | 2017.0 | 1400000 | fullstack engineer | 2019 | 1000K-1500K | |

In [45]:

```
distinct_job_pos_per_email = data.groupby(['email_hash'])['job_position'].nunique().reset_index()
distinct_job_pos_per_email.rename(columns={'job_position': 'distinct_job_pos_per_email'}, inplace=True)

data = data.merge(distinct_job_pos_per_email, on=['email_hash'], how = "left")
data.head()
```

Out[45]:

| | company_hash | email_hash | orgyear | ctc | job_position | ctc_updated_year | ctc_binned | com |
|---|---|---|---|---|---|---|---|---|
| **0** | atrgxnnt xzaxv | 6de0a4417d18ab14334c3f43397fc13b30c35149d70c05... | 2016.0 | 1100000 | other | 2020 | 1000K-1500K | |
| **1** | qtrxvzwt xzegwgbb rxbxnta | b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10... | 2018.0 | 449999 | fullstack engineer | 2019 | 100K-500K | |
| **2** | ojzwnvwnxw vx | 4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9... | 2015.0 | 2000000 | backend engineer | 2020 | 1500K-10000K | |
| **3** | ngpgutaxv | effdede7a2e7c2af664c8a31d9346385016128d66bbc58... | 2017.0 | 700000 | backend engineer | 2019 | 500K-1000K | |
| **4** | qxen sqghu | 6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520... | 2017.0 | 1400000 | fullstack engineer | 2019 | 1000K-1500K | |

In [46]:

```
number_times_person_ctc_updated_in_same_company = data.groupby(['company_hash','email_hash'])['orgyear'].nunique().reset_index
number_times_person_ctc_updated_in_same_company.rename(columns={'orgyear': 'ctc_updated_cnt_same_company'}, inplace=True)

data = data.merge(number_times_person_ctc_updated_in_same_company, on=['company_hash','email_hash'], how = "left")
data.head()
```

Out[46]:

| | company_hash | email_hash | orgyear | ctc | job_position | ctc_updated_year | ctc_binned | com |
|---|---|---|---|---|---|---|---|---|
| 0 | atrgxnnt xzaxv | 6de0a4417d18ab14334c3f43397fc13b30c35149d70c05... | 2016.0 | 1100000 | other | 2020 | 1000K-1500K | |
| 1 | qtrxvzwt xzegwgbb rxbxnta | b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10... | 2018.0 | 449999 | fullstack engineer | 2019 | 100K-500K | |
| 2 | ojzwnvwnxw vx | 4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9... | 2015.0 | 2000000 | backend engineer | 2020 | 1500K-10000K | |
| 3 | ngpgutaxv | effdede7a2e7c2af664c8a31d9346385016128d66bbc58... | 2017.0 | 700000 | backend engineer | 2019 | 500K-1000K | |
| 4 | qxen sqghu | 6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520... | 2017.0 | 1400000 | fullstack engineer | 2019 | 1000K-1500K | |

## New Feature Creation -Years of Experience

In [47]:
```
current_year = datetime.datetime.now().year
current_year
```

Out[47]: 2024

In [48]:
```
data['years_of_experience'] = current_year - data['orgyear']
```

In [49]:
```
# Considering -ve and more then 100 years of experience as Invalid(0)
data.loc[ (data['years_of_experience'] < 0) | (data['years_of_experience'] > 100 ),'years_of_experience'] = 0
```

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

Out[50]:

| | company_hash | email_hash | orgyear | ctc | job_position | ctc_updated_year | ctc_binned | comp |
|---|---|---|---|---|---|---|---|---|
| **0** | atrgxnnt xzaxv | 6de0a4417d18ab14334c3f43397fc13b30c35149d70c05... | 2016.0 | 1100000 | other | 2020 | 1000K-1500K | |
| **1** | qtrxvzwt xzegwgbb rxbxnta | b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10... | 2018.0 | 449999 | fullstack engineer | 2019 | 100K-500K | |
| **2** | ojzwnvwnxw vx | 4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9... | 2015.0 | 2000000 | backend engineer | 2020 | 1500K-10000K | |
| **3** | ngpgutaxv | effdede7a2e7c2af664c8a31d9346385016128d66bbc58... | 2017.0 | 700000 | backend engineer | 2019 | 500K-1000K | |
| **4** | qxen sqghu | 6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520... | 2017.0 | 1400000 | fullstack engineer | 2019 | 1000K-1500K | |

# 3. Univariate and Bivariate Analysis

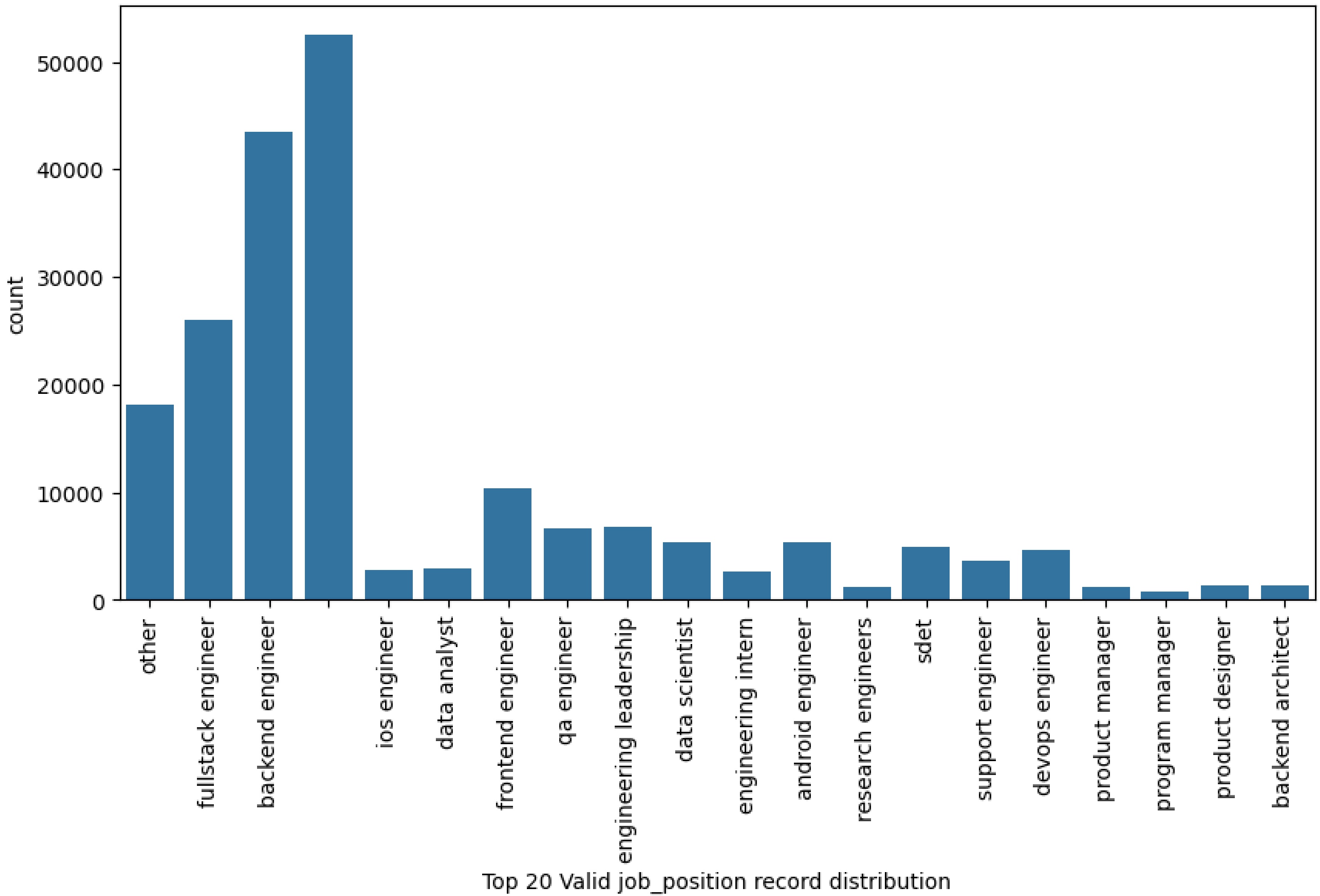
## 3.1 Univariate Analysis

In [51]:
```python
def get_top_20_value_records(data,feature_name):
    top_10_names = data[feature_name].value_counts().index[:20]
    return data[data[feature_name].isin(top_10_names)]
```

In [52]:
```python
data.info()
```

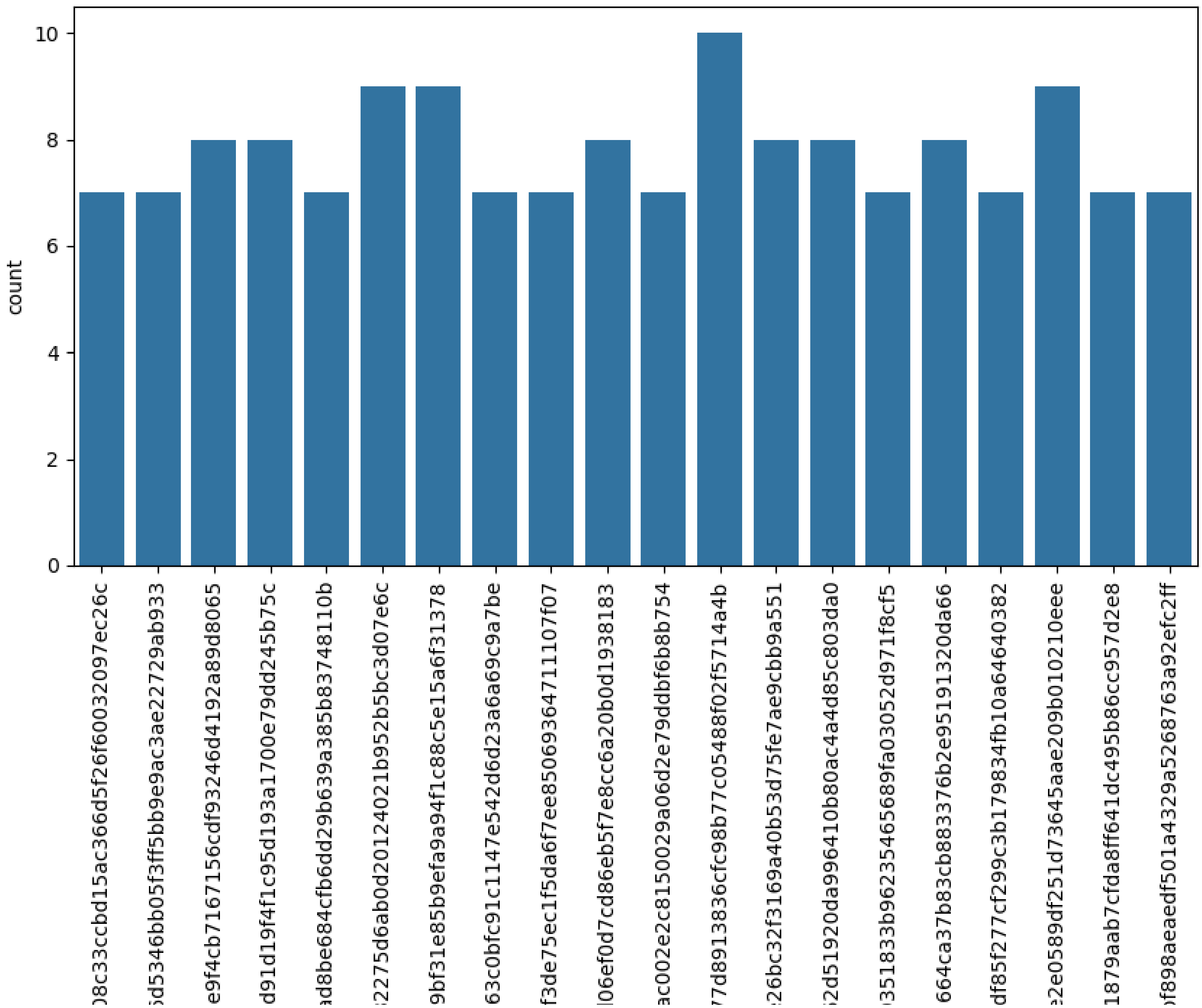
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205809 entries, 0 to 205808
Data columns (total 14 columns):
 #   Column                               Non-Null Count   Dtype   
---  ------                               --------------   -----   
 0   company_hash                         205809 non-null  object  
 1   email_hash                           205809 non-null  object  
 2   orgyear                              205809 non-null  float64 
 3   ctc                                  205809 non-null  int64   
 4   job_position                         205809 non-null  object  
 5   ctc_updated_year                     205809 non-null  int32   
 6   ctc_binned                           205809 non-null  category
 7   company_median_CTC                   205809 non-null  float64 
 8   email_avg_CTC                        205809 non-null  float64 
 9   job_pos_avg_CTC_accross_all_companies  205809 non-null  float64 
 10  job_pos_avg_CTC_accross_same_company  205809 non-null  float64 
 11  distinct_job_pos_per_email           205809 non-null  int64   
 12  ctc_updated_cnt_same_company         205809 non-null  int64   
 13  years_of_experience                  205809 non-null  float64 
dtypes: category(1), float64(6), int32(1), int64(3), object(3)
memory usage: 19.8+ MB
```

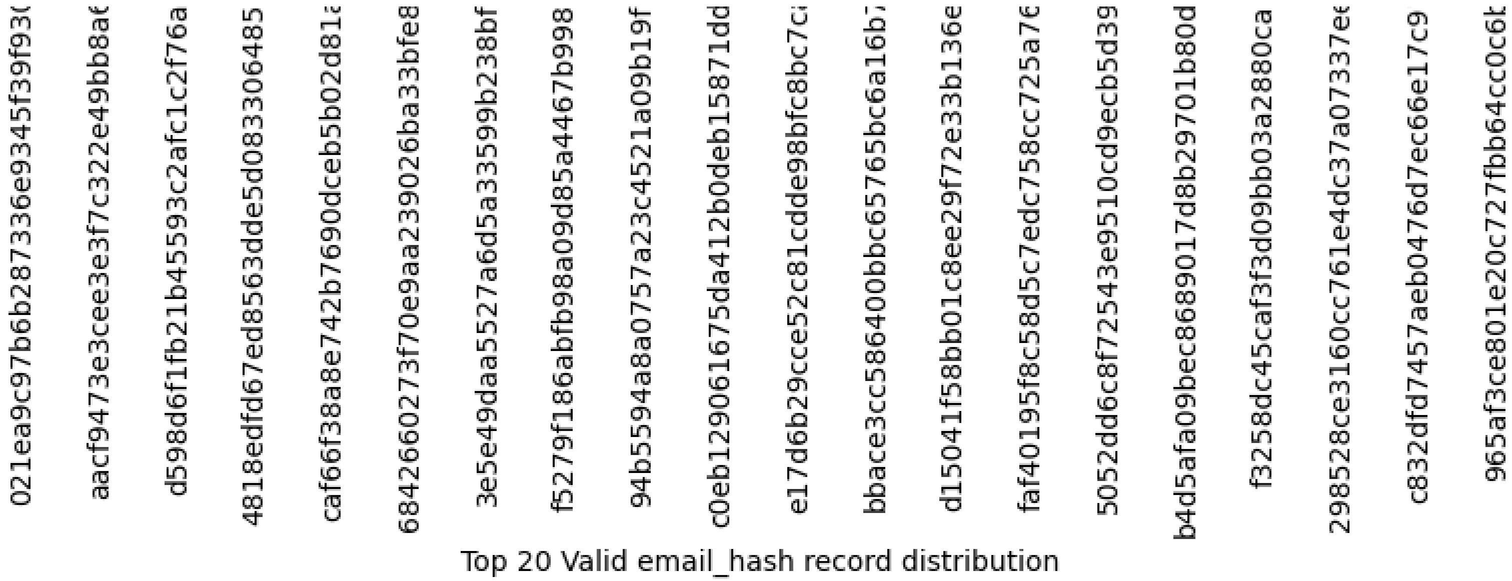
In [53]:
```
plt.figure(figsize=(10, 5))
top_20_jobs_data =  get_top_20_value_records(data,'job_position')
sns.countplot(data=top_20_jobs_data, x='job_position')
plt.xlabel('Top 20 Valid job_position record distribution')
plt.xticks(rotation=90)
plt.show()
```



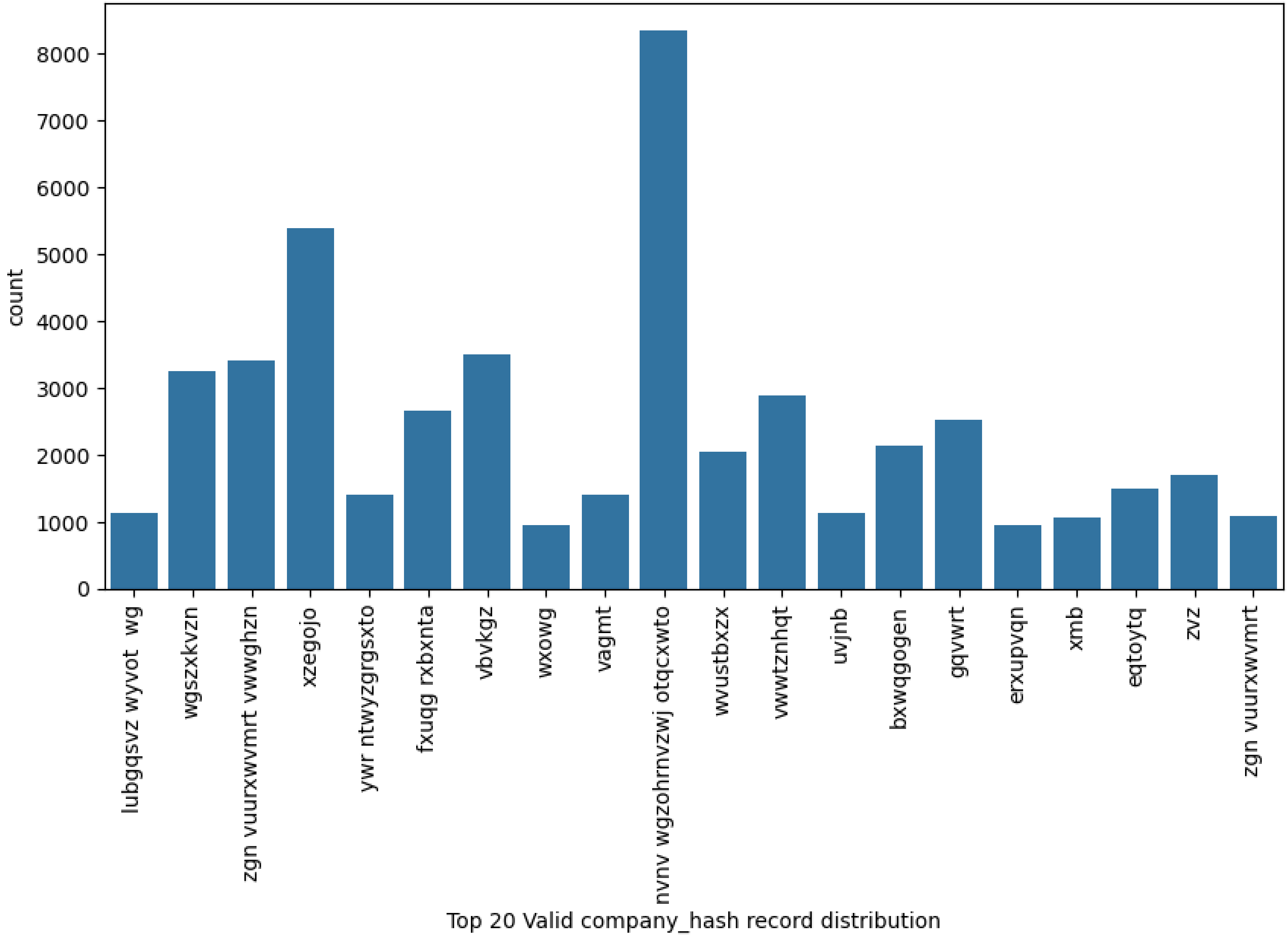

In [54]:
```
plt.figure(figsize=(10, 5))
top_20_email_data =  get_top_20_value_records(data,'email_hash')
sns.countplot(data=top_20_email_data, x='email_hash')
plt.xlabel('Top 20 Valid email_hash record distribution')
```

```
plt.xticks(rotation=90)
plt.show()
```


count
0
2
4
6
8
10

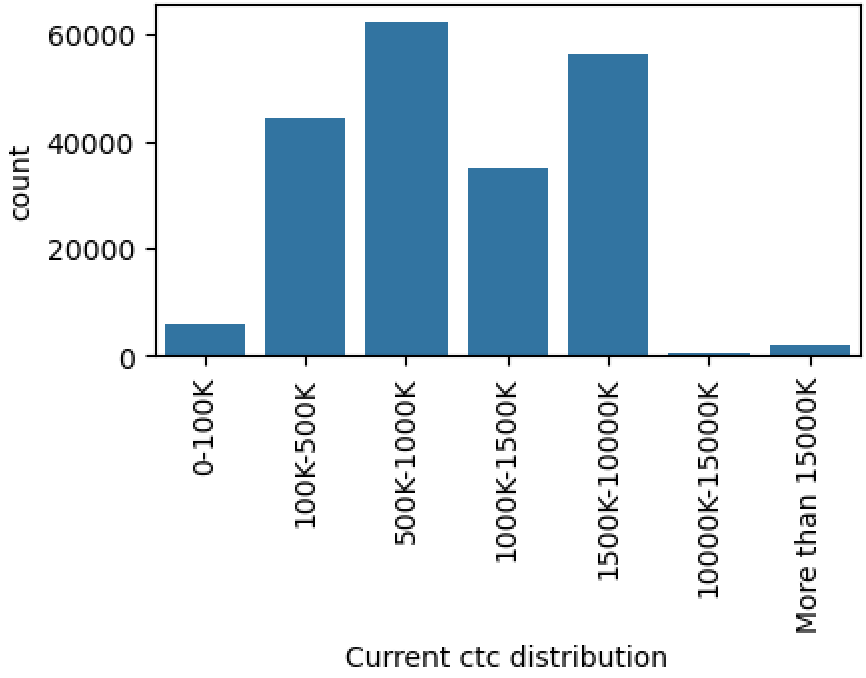



In [55]:
```
plt.figure(figsize=(10, 5))
top_20_company_data =  get_top_20_value_records(data,'company_hash')
sns.countplot(data=top_20_company_data, x='company_hash')
plt.xlabel('Top 20 Valid company_hash record distribution')
plt.xticks(rotation=90)
plt.show()
```



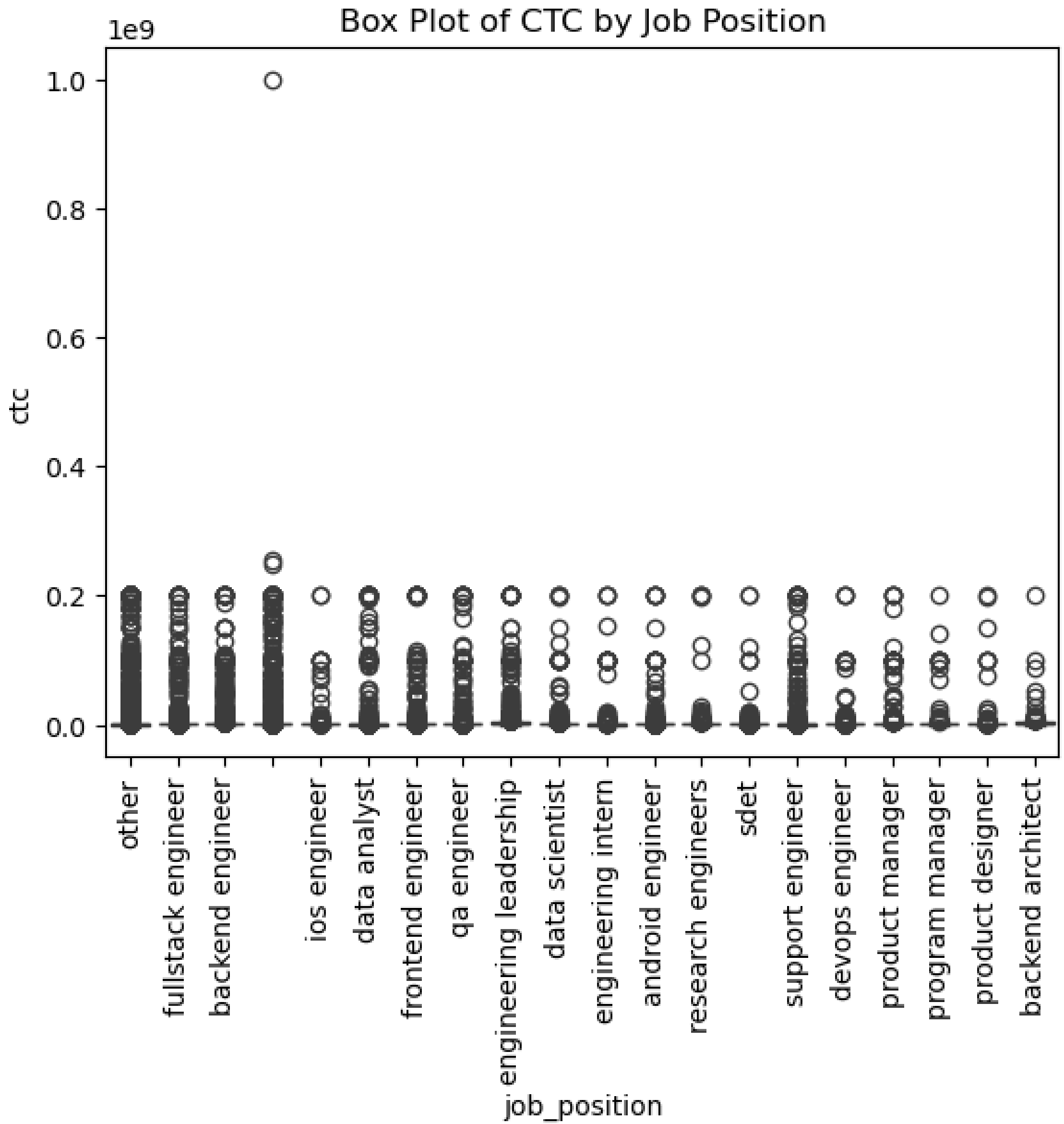

In [56]:
```
plt.figure(figsize=(10, 5))
plt.subplot(224)
```

```
sns.countplot(data=data, x='ctc_binned')
plt.xlabel('Current ctc distribution')
plt.xticks(rotation=90)
plt.show()
```



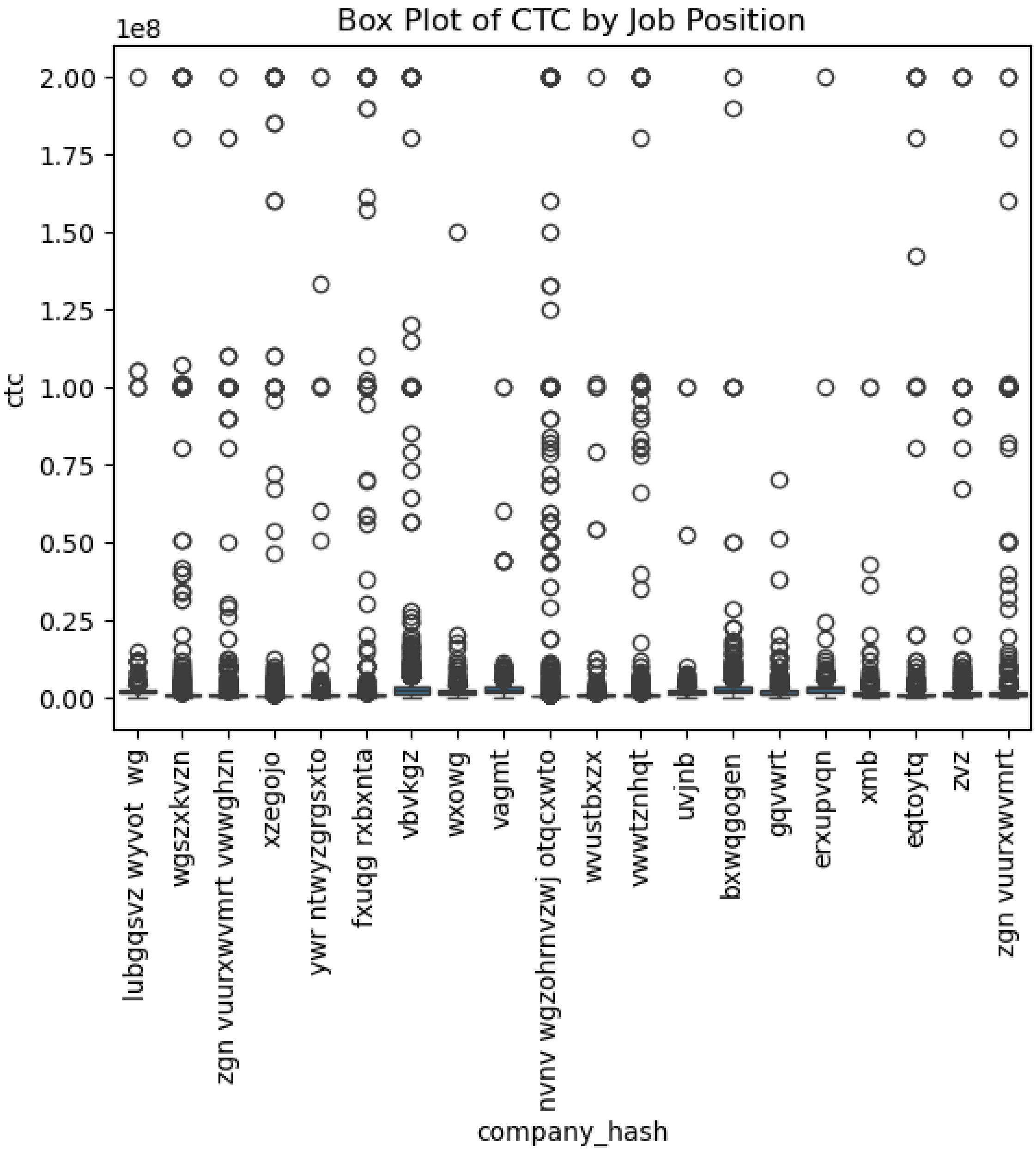

## 3.2 Bivariate Analysis

In [57]:
```
sns.boxplot(x='job_position', y='ctc', data=top_20_jobs_data)
plt.title('Box Plot of CTC by Job Position')
plt.xticks(rotation=90)
plt.show()
```




In [58]:

```
sns.boxplot(x='company_hash', y='ctc', data=top_20_company_data)
plt.title('Box Plot of CTC by Job Position')
plt.xticks(rotation=90)
plt.show()
```




```
In [59]: num_column = ['orgyear', 'ctc','ctc_updated_year', 'company_median_CTC', 'email_avg_CTC',
                       'job_pos_avg_CTC_accross_all_companies','job_pos_avg_CTC_accross_same_company', 'distinct_job_pos_per_email',
```

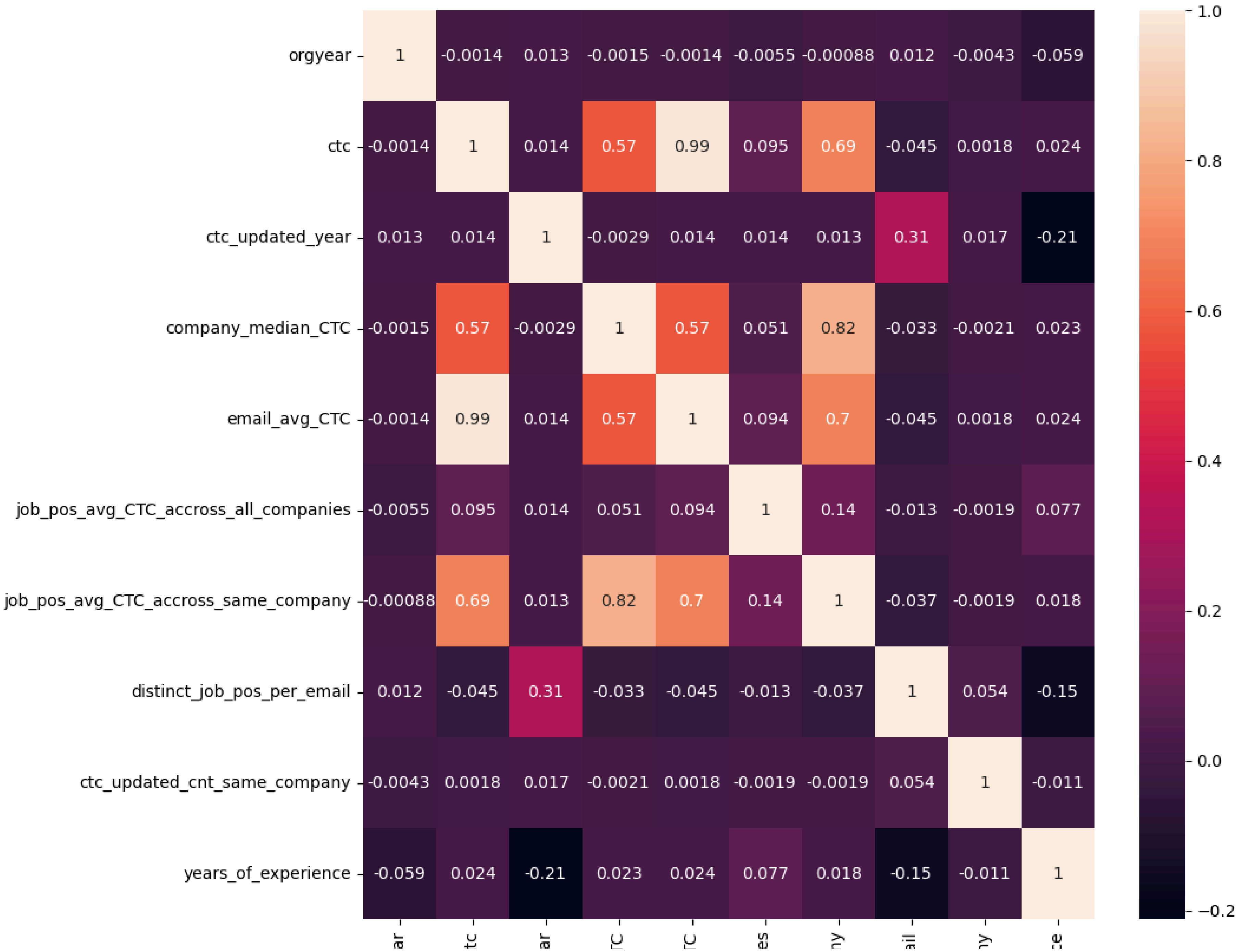
```
                 'ctc_updated_cnt_same_company', 'years_of_experience']
```

In [60]:
```
data[num_column].corr()
```

Out[60]:

| | orgyear | ctc | ctc_updated_year | company_median_CTC | email_avg_CTC | job_pos_avg_CTC_acc |
|---|---|---|---|---|---|---|
| **orgyear** | 1.000000 | -0.001429 | 0.012598 | -0.001486 | -0.001449 | |
| **ctc** | -0.001429 | 1.000000 | 0.014269 | 0.572683 | 0.992144 | |
| **ctc_updated_year** | 0.012598 | 0.014269 | 1.000000 | -0.002918 | 0.013806 | |
| **company_median_CTC** | -0.001486 | 0.572683 | -0.002918 | 1.000000 | 0.574617 | |
| **email_avg_CTC** | -0.001449 | 0.992144 | 0.013806 | 0.574617 | 1.000000 | |
| **job_pos_avg_CTC_accross_all_companies** | -0.005505 | 0.094533 | 0.013959 | 0.051346 | 0.093688 | |
| **job_pos_avg_CTC_accross_same_company** | -0.000879 | 0.694990 | 0.012992 | 0.824015 | 0.695489 | |
| **distinct_job_pos_per_email** | 0.011753 | -0.045036 | 0.309133 | -0.033213 | -0.045393 | |
| **ctc_updated_cnt_same_company** | -0.004277 | 0.001808 | 0.016632 | -0.002052 | 0.001823 | |
| **years_of_experience** | -0.058813 | 0.024253 | -0.211845 | 0.023333 | 0.024466 | |

In [61]:
```
plt.figure(figsize = (10,10))
sns.heatmap(data[num_column].corr() , annot=True)
plt.show()
```


orgyear
ctc
ctc_updated_year
company_median_CTC
email_avg_CTC
job_pos_avg_CTC_accross_all_companies
job_pos_avg_CTC_accross_same_company
distinct_job_pos_per_email
ctc_updated_cnt_same_company
years_of_experience
1 -0.0014 0.013 -0.0015 -0.0014 -0.0055 -0.00088 0.012 -0.0043 -0.059
-0.0014 1 0.014 0.57 0.99 0.095 0.69 -0.045 0.0018 0.024
0.013 0.014 1 -0.0029 0.014 0.014 0.013 0.31 0.017 -0.21
-0.0015 0.57 -0.0029 1 0.57 0.051 0.82 -0.033 -0.0021 0.023
-0.0014 0.99 0.014 0.57 1 0.094 0.7 -0.045 0.0018 0.024
-0.0055 0.095 0.014 0.051 0.094 1 0.14 -0.013 -0.0019 0.077
-0.00088 0.69 0.013 0.82 0.7 0.14 1 -0.037 -0.0019 0.018
0.012 -0.045 0.31 -0.033 -0.045 -0.013 -0.037 1 0.054 -0.15
-0.0043 0.0018 0.017 -0.0021 0.0018 -0.0019 -0.0019 0.054 1 -0.011
-0.059 0.024 -0.21 0.023 0.024 0.077 0.018 -0.15 -0.011 1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

| orgye | c | ctc_updated_ye | company_median_CT | email_avg_CT | job_pos_avg_CTC_accross_all_compani | job_pos_avg_CTC_accross_same_compa | distinct_job_pos_per_em | ctc_updated_cnt_same_compa | years_of_experien |
|---|---|---|---|---|---|---|---|---|---|

# 4. Manual Clustering

- on the basis of learner's company, job position and years of experience

In [62]:
```
data_agg = data.groupby(['company_hash', 'job_position', 'years_of_experience'])[['ctc']].aggregate( avg_ctc_by_company_dept_e
                                                                                                        median_ctc_by_company_dept_exp = ('ctc'
                                                                                                        max_ctc_by_company_dept_exp = ('ctc','m
                                                                                                        min_ctc_by_company_dept_exp = ('ctc','m
                                                                                                        count_ctc_by_company_dept_exp = ('ctc',
                                                                                                       ).reset_index()
data_agg
```

Out[62]:

| | company_hash | job_position | years_of_experience | avg_ctc_by_company_dept_exp | median_ctc_by_company_dept_exp | max_ctc_by_co |
|---|---|---|---|---|---|---|
| **0** | | | 2.0 | 6.660000e+07 | 66600000.0 | |
| **1** | | | 3.0 | 9.000000e+05 | 900000.0 | |
| **2** | | | 4.0 | 5.666667e+05 | 600000.0 | |
| **3** | | | 5.0 | 1.133333e+06 | 500000.0 | |
| **4** | | | 6.0 | 6.976665e+05 | 450000.0 | |
| **...** | ... | ... | ... | ... | ... | |
| **113157** | zz | other | 11.0 | 1.370000e+06 | 1370000.0 | |
| **113158** | zzb ztdnstz vacxogqj ucn rna | | 7.0 | 6.000000e+05 | 600000.0 | |
| **113159** | zzb ztdnstz vacxogqj ucn rna | fullstack engineer | 7.0 | 6.000000e+05 | 600000.0 | |
| **113160** | zzgato | | 10.0 | 1.300000e+05 | 130000.0 | |
| **113161** | zzzbzb | other | 34.0 | 7.200000e+05 | 720000.0 | |

113162 rows × 8 columns

In [63]:

```
data = data.merge(data_agg , on = ['company_hash','job_position','years_of_experience'] , how = "left")
data.head()
```

Out[63]:

| | company_hash | email_hash | orgyear | ctc | job_position | ctc_updated_year | ctc_binned | comp |
|---|---|---|---|---|---|---|---|---|
| 0 | atrgxnnt xzaxv | 6de0a4417d18ab14334c3f43397fc13b30c35149d70c05... | 2016.0 | 1100000 | other | 2020 | 1000K-1500K | |
| 1 | qtrxvzwt xzegwgbb rxbxnta | b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10... | 2018.0 | 449999 | fullstack engineer | 2019 | 100K-500K | |
| 2 | ojzwnvwnxw vx | 4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9... | 2015.0 | 2000000 | backend engineer | 2020 | 1500K-10000K | |
| 3 | ngpgutaxv | effdede7a2e7c2af664c8a31d9346385016128d66bbc58... | 2017.0 | 700000 | backend engineer | 2019 | 500K-1000K | |
| 4 | qxen sqghu | 6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520... | 2017.0 | 1400000 | fullstack engineer | 2019 | 1000K-1500K | |

In [64]:
```
# creating some flags showing learners with CTC greater than the Average of their Company’s department having same Years of Ex
# Doing above analysis at Company & Job Position level. Name that flag Class with values [1,2,3]
# Repeating the same analysis at the Company level. Name that flag Tier with values [1,2,3]
```

In [65]:
```
def flag_des_rule(row):
    if row['ctc'] > row['avg_ctc_by_company_dept_exp']:
        return 1
    elif row['ctc'] < row['avg_ctc_by_company_dept_exp']:
        return 3
    else:
        return 2
data['flag_designation'] = data.apply(flag_des_rule, axis=1)
```

In [66]:
```
def flag_class_rule(row):
    if row['ctc'] > row['job_pos_avg_CTC_accross_same_company']:
        return 1
    elif row['ctc'] < row['job_pos_avg_CTC_accross_same_company']:
        return 3
    else:
```

```
        return 2
data['flag_class'] = data.apply(flag_class_rule, axis=1)
```

In [67]:
```
company_avg_ctc = data.groupby('company_hash')['ctc'].mean().reset_index()
company_avg_ctc.rename(columns={'ctc': 'company_mean_ctc'}, inplace=True)

# Merge average CTC back to original dataframe
data = data.merge(company_avg_ctc, on='company_hash' , how ="left")
```

In [68]:
```
def flag_tier_rule(row):
    if row['ctc'] > row['company_mean_ctc']:
        return 1
    elif row['ctc'] < row['company_mean_ctc']:
        return 3
    else:
        return 2
data['flag_tier'] = data.apply(flag_tier_rule, axis=1)
```

In [69]:
```
data
```

Out[69]:

| | company_hash | email_hash | orgyear | ctc | job_position | ctc_updated_year | ctc_binned |
|---|---|---|---|---|---|---|---|
| **0** | atrgxnnt xzaxv | 6de0a4417d18ab14334c3f43397fc13b30c35149d70c05... | 2016.0 | 1100000 | other | 2020 | 1000K-1500K |
| **1** | qtrxvzwt xzegwgbb rxbxnta | b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10... | 2018.0 | 449999 | fullstack engineer | 2019 | 100K-500K |
| **2** | ojzwnvwnxw vx | 4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9... | 2015.0 | 2000000 | backend engineer | 2020 | 1500K-10000K |
| **3** | ngpgutaxv | effdede7a2e7c2af664c8a31d9346385016128d66bbc58... | 2017.0 | 700000 | backend engineer | 2019 | 500K-1000K |
| **4** | qxen sqghu | 6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520... | 2017.0 | 1400000 | fullstack engineer | 2019 | 1000K-1500K |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **205804** | vuurt xzw | 70027b728c8ee901fe979533ed94ffda97be08fc23f33b... | 2008.0 | 220000 | | 2019 | 100K-500K |
| **205805** | husqvawgb | 7f7292ffad724ebbe9ca860f515245368d714c84705b42... | 2017.0 | 500000 | | 2020 | 100K-500K |
| **205806** | vwwgrxnt | cb25cc7304e9a24facda7f5567c7922ffc48e3d5d6018c... | 2021.0 | 700000 | | 2021 | 500K-1000K |
| **205807** | zgn vuurxwvmrt | fb46a1a2752f5f652ce634f6178d0578ef6995ee59f6c8... | 2019.0 | 5100000 | | 2019 | 1500K-10000K |
| **205808** | bgqsvz onvzrtj | 0bcfc1d05f2e8dc4147743a1313aa70a119b41b30d4a1f... | 2014.0 | 1240000 | | 2016 | 1000K-1500K |

205809 rows × 23 columns

## 4.1 Question and Answers of Manual Clustering

In [70]:
```
# Top 10 employees (earning more than most of the employees in the company) - Tier 1
top_employees = data.loc[data['flag_tier'] == 1, :]
```

```
top_10_employees = top_employees.sort_values(by ='ctc' , ascending  = False)[:10]
top_10_employees['email_hash']
```

Out[70]:

```
117626    5b4bed51797140db4ed52018a979db1e34cee49e27b488...
12646     0f0d1bf4233dadef653775c3f981f0ccde1dc20df36c43...
126131    7683974378d0f5bacf95632f130a60cfa3ca39e368eec9...
27075     0f0e7a9db34d1317498d9378a1bd0150bfa022ac6ba3a0...
17835     a35a5abbe9fb056421bdd9aca4440acfb93e37c823564d...
78783     76708a11cb61a030ff3da827b0fd19aff536c3793c1816...
7345      68aa38470922a03f6022280b2a13c6f5ab6a717f70c77a...
61281     2f9a4241053f76b2f8c50ea593a90586d38b3f0e08c141...
61288     3c85c094eaeb923add569ed91b8fbd6f8d8e8194dbaff5...
2279      1f8e216b2328e7764f79dbd66deaf008b409870ae992fe...
Name: email_hash, dtype: object
```

In [71]:

```
# Top 10 employees of data science in each company earning more than their peers - Class 1
top_employees = data.loc[ (data['flag_class'] == 1) & (data['job_position'].str.contains('data scientist') ), :]
top_10_employees = top_employees.sort_values(by ='ctc' , ascending  = False)[:10]
top_10_employees['email_hash']
```

Out[71]:

```
31297     bd222ea783ee372da4e0ad60fdccec0b8f37999a032025...
52818     268a5aa92f0b6d0c675fc9cc1e300eb0c5930a3a139a23...
836       cda8d723438e81185d2ee8c348870a4612eea974cdb2db...
122734    6b6dd66bae787dd4dd417e1777f8ea5a057257e9019995...
151477    6ad86d120e39db485331f9a0b2b1f15ce2a7bdaee778ab...
143283    4ddef8762b7585c6ee7b8c06834778f3aa00eb3be312b0...
57755     75f5b46d47310c3923e93329a62a1aa78d478803f0a685...
152656    544e75b477f8644eb71281133c62c19732547837e80e51...
32753     15adaeb2eef9c0ee8a0f18e189bf426be390f5d1e911fd...
16678     3c64901d83458f3b7b8eed6fb529ee3a4c14d49339c398...
Name: email_hash, dtype: object
```

In [72]:

```
# Bottom 10 employees of data science in each company earning less than their peers - Class 3
bottom_employees = data.loc[ (data['flag_class'] == 3) & (data['job_position'].str.contains('data scientist') ), :]
bottom_10_employees = bottom_employees.sort_values(by ='ctc' , ascending  = True)[:10]
bottom_10_employees['email_hash']
```

Out[72]:
```
8705      690f6fdab1ab7514a6a9325ebd6cfe910dbf12d46b6fde...
10835     8001bc017fbe95541d23f5780c3edb988b7d9b2225e39e...
51030     bd9c04a574090e05b366a81cdb2f3f565d0c60fa8b1647...
136954    e374eea75640881206a21894f69190138c2c0535277dc1...
24107     ab2dc9db23c3104f0b6b3dbd4cdd5bfb9e5829b8b7943d...
9403      3175d03fd4618eb293d6f5a1d13d42a0c79f68e9acaaa3...
24104     3675f79c7e05de96ccf189c818b84b487cb1aa3f6b80e8...
31750     fb64af615420e06d46a1965f59068b34460fb3cbe70541...
168145    8274b3188470cd1c4914e7face490111e27f239457e62d...
82800     3cc0c85d198d0e56a4cdefb6496333f59b97f87c293262...
Name: email_hash, dtype: object
```

In [73]:
```python
# Bottom 10 employees (earning less than most of the employees in the company)- Tier 3
bottom_employees = data.loc[ data['flag_tier'] == 3, :]
bottom_10_employees = bottom_employees.sort_values(by ='ctc' , ascending  = True)[:10]
bottom_10_employees['email_hash']
```

Out[73]:
```
135421    3505b02549ebe2c95840ac6f0a35561a3b4cbe4b79cdb1...
118226    f2b58aeed3c074652de2cfd3c0717a5d21d6fbcf342a78...
114157    23ad96d6b6f1ecf554a52f6e9b61677c7d73d8a409a143...
184918    b8a0bb340583936b5a7923947e9aec21add5ebc50cd60b...
116938    f7e5e788676100d7c4146740ada9e2f8974defc01f571d...
150664    9af3dca6c9d705d8d42585ccfce2627f00e1629130d14e...
171173    80ba0259f9f59034c4927cf3bd38dc9ce2eb60ff18135b...
99417     b995d7a2ae5c6f8497762ce04dc5c04ad6ec734d70802a...
77157     f0f2005505c707dbdd2c86ca1587c26f822a004e86a8ec...
159510    afa111053d280ef49ea791a3b5f7d171f961f4bd8ec724...
Name: email_hash, dtype: object
```

In [74]:
```python
# Top 10 employees in each company - X department - having 5/6/7 years of experience earning more than their peers - Tier X
top_employees = data.loc[ (data['flag_tier'] == 1) &
    ( (data['years_of_experience'] == 5)|(data['years_of_experience'] == 6)|(data['years_of_experience'] == 7) ),:]
top_10_employees = top_employees.sort_values(by ='ctc' , ascending  = False)[:10]
top_10_employees['email_hash']
```

Out[74]:
```
117626    5b4bed51797140db4ed52018a979db1e34cee49e27b488...
104891    1b95e7ba0ee82100ca5a034239fa0203a1bec14280b82a...
134579    1b95e7ba0ee82100ca5a034239fa0203a1bec14280b82a...
60988     76fec55391d520a956adabf982d88faf6842b6188b901f...
60349     38e99d23328a60acdf25e8c4fa7616661093fd4874151a...
9609      9e785d33821db67c01becc1c36f901d79d3142c1d13bd8...
1083      a071c4cd6d423e8d1841ba6133e6c4684f4eaba7dc1526...
22973     634fd283565b8954513a6ad0e47cedb0fa8847923149fb...
12601     89f343bf01094accb8b0b2c799499daf6bf881321db2e4...
82601     2311bf023218afe93d650cac03abb7a40f7fa55c08d260...
Name: email_hash, dtype: object
```

In [75]:
```
# Top 10 companies (based on their CTC)
top_company = data[['company_hash','company_mean_ctc']].drop_duplicates()
top_10_company = top_company.sort_values(by ='company_mean_ctc' , ascending  = False)[:10]
top_10_company['company_hash']
```

Out[75]:
```
72824                          whmxw rgsxwo uqxcvnt rxbxnta
3301                       aveegaxr xzntqzvnxgzvr hzxctqoxnj
2517                                      zxoyvzn wgbuhntqo
26292     vooxontvoj cxqnhvr onveexzs uqxcvnt ogrhnxgzo
19594                              ctqexohayv ntwyzgrgsxto
29964                                     durgfxk ogrhnxgzo
139354                               nhqvmxn  vx vooxonvzno
12603                                              omx rxet
44172                                    gqmxn ogenfvqt xzw
487                                      xfgqp ntwyzgrgsxto
Name: company_hash, dtype: object
```

In [76]:
```
# Top 2 positions in every company (based on their CTC)
top_positions = data.sort_values(['company_hash', 'ctc'], ascending=[True, False]).groupby('company_hash').head(2)
top_positions
```

Out[76]:

| | company_hash | email_hash | orgyear | ctc | job_position | ctc_updated_year | ctc_binne |
|---|---|---|---|---|---|---|---|
| **1115** | | 8fe09b732fe2e5b66c14904fd02ff89fb54f458465ac1e... | 2022.0 | 66600000 | | 2020 | More tha 15000 |
| **68706** | | 8fe09b732fe2e5b66c14904fd02ff89fb54f458465ac1e... | 2022.0 | 66600000 | database administrator | 2020 | More tha 15000 |
| **2940** | 0 | e80f7c9c26012bfdeca551e2b8642a93e45939d3d677c5... | 2020.0 | 100000 | | 2020 | 0-100 |
| **16824** | 0 | e80f7c9c26012bfdeca551e2b8642a93e45939d3d677c5... | 2020.0 | 100000 | other | 2020 | 0-100 |
| **197509** | 0000 | b3f3bb98cbca4b1ce5dfd5abb4e500ce6f6b66288a5202... | 2017.0 | 300000 | other | 2020 | 100K-500 |
| **...** | ... | ... | ... | ... | ... | ... | |
| **14670** | zz | d6923a6f81c7b36615d9f14349fe01aec442029b2c502f... | 2009.0 | 500000 | | 2021 | 100K-500 |
| **72983** | zzb ztdnstz vacxogqj ucn rna | ca8935e2314a1bac3947e60bbd2ee10524112898da29eb... | 2017.0 | 600000 | fullstack engineer | 2021 | 500k 1000 |
| **146611** | zzb ztdnstz vacxogqj ucn rna | ca8935e2314a1bac3947e60bbd2ee10524112898da29eb... | 2017.0 | 600000 | | 2021 | 500k 1000 |
| **117015** | zzgato | d421e52125f8057c65fa554752be03b056221c8590ff26... | 2014.0 | 130000 | | 2017 | 100K-500 |
| **15838** | zzzbzb | 3d4fedde22283c75bb84220e962360c352991c3054839c... | 1990.0 | 720000 | other | 2020 | 500k 1000 |

51106 rows × 23 columns

# OBservation:

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

['5b4bed51797140db4ed52018a979db1e34cee49e27b4885c3fdfacea9f8144f6',
'0f0d1bf4233dadef653775c3f981f0ccde1dc20df36c432e934262407f79a7d6',
'7683974378d0f5bacf95632f130a60cfa3ca39e368eec9b0d50d7719c193d758',
'0f0e7a9db34d1317498d9378a1bd0150bfa022ac6ba3a02feaa740b2ae6f9927',
'a35a5abbe9fb056421bdd9aca4440acfb93e37c823564d952a50c40583920598',
'76708a11cb61a030ff3da827b0fd19aff536c3793c181689535e04c729dcfddc',
'68aa38470922a03f6022280b2a13c6f5ab6a717f70c77a0f875262a742947ce3',
'2f9a4241053f76b2f8c50ea593a90586d38b3f0e08c141d56edad2b811c0829c',
'3c85c094eaeb923add569ed91b8fbd6f8d8e8194dbaff5614338dbafb4c88074',
'1f8e216b2328e7764f79dbd66deaf008b409870ae992fe6eba3f914c6

## Top 10 employees of data science in each company earning more than their peers - Class 1

['bd222ea783ee372da4e0ad60fdccec0b8f37999a032025d8a83d9864bdb975ec',
'268a5aa92f0b6d0c675fc9cc1e300eb0c5930a3a139a23bc924d036bc6f3e3a5',
'cda8d723438e81185d2ee8c348870a4612eea974cdb2dbde0e99895d201a88ee',
'6b6dd66bae787dd4dd417e1777f8ea5a057257e9019995056ab8ad566995aca2',
'6ad86d120e39db485331f9a0b2b1f15ce2a7bdaee778ab0e1afea8f486d84eb8',
'4ddef8762b7585c6ee7b8c06834778f3aa00eb3be312b05e567e74969695264c',
'75f5b46d47310c3923e93329a62a1aa78d478803f0a685fec70995f380c0f5bc',
'544e75b477f8644eb71281133c62c19732547837e80e51060928888e0e7f4e90',
'15adaeb2eef9c0ee8a0f18e189bf426be390f5d1e911fdb086aa4e2b567b35ab',
'3c64901d83458f3b7b8eed6fb529ee3a4c14d49339c39810aa751621c

## Bottom 10 employees of data science in each company earning less than their peers - Class 3

['690f6fdab1ab7514a6a9325ebd6cfe910dbf12d46b6fdef54f0e06c098a66dcd',
'8001bc017fbe95541d23f5780c3edb988b7d9b2225e39eec4629758f1257d2fe',
'bd9c04a574090e05b366a81cdb2f3f565d0c60fa8b1647949a3cc44ad282c485',
'e374eea75640881206a21894f69190138c2c0535277dc1909cec80f734e9a08f',
'ab2dc9db23c3104f0b6b3dbd4cdd5bfb9e5829b8b7943d5185fb535fe7c6b7c1',

'3175d03fd4618eb293d6f5a1d13d42a0c79f68e9acaaa31e0f4da21ae8f1b6f0',
'3675f79c7e05de96ccf189c818b84b487cb1aa3f6b80e886ba5cd94a5d1e79e8',
'fb64af615420e06d46a1965f59068b34460fb3cbe70541c6c1852e8458396a62',
'8274b3188470cd1c4914e7face490111e27f239457e62d7edb2ee1353e8cb18b',
'3cc0c85d198d0e56a4cdefb6496333f59b97f87c2932629c2de53d4cd

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

['3505b02549ebe2c95840ac6f0a35561a3b4cbe4b79cdb15d794a6ac066b66644',
'f2b58aeed3c074652de2cfd3c0717a5d21d6fbcf342a786928c5fd38c860fa45',
'23ad96d6b6f1ecf554a52f6e9b61677c7d73d8a409a143cee85eda1557f516c8',
'b8a0bb340583936b5a7923947e9aec21add5ebc50cd60bf6953ea67074932d41',
'f7e5e788676100d7c4146740ada9e2f8974defc01f571d34a3ff1ccf6b68c0a8',
'9af3dca6c9d705d8d42585ccfce2627f00e1629130d14ef814d03bd2ac256596',
'80ba0259f9f59034c4927cf3bd38dc9ce2eb60ff18135bf9012feac4c1169c23',
'b995d7a2ae5c6f8497762ce04dc5c04ad6ec734d70802a94f83e17431884e907',
'f0f2005505c707dbdd2c86ca1587c26f822a004e86a8ec9caebc6145336fe83d',
'afa111053d280ef49ea791a3b5f7d171f961f4bd8ec7243613aa944f3

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

['5b4bed51797140db4ed52018a979db1e34cee49e27b4885c3fdfacea9f8144f6',
'1b95e7ba0ee82100ca5a034239fa0203a1bec14280b82a5cc81a192bb7c82df3',
'1b95e7ba0ee82100ca5a034239fa0203a1bec14280b82a5cc81a192bb7c82df3',
'76fec55391d520a956adabf982d88faf6842b6188b901feddff30a2a307d0c66',
'38e99d23328a60acdf25e8c4fa7616661093fd4874151a9a58ea0f0650e02823',
'9e785d33821db67c01becc1c36f901d79d3142c1d13bd84cf8aa670f30717e20',
'a071c4cd6d423e8d1841ba6133e6c4684f4eaba7dc15261313f206d04e640ebd',
'634fd283565b8954513a6ad0e47cedb0fa8847923149fbae5f3d3889075837b6',

'89f343bf01094accb8b0b2c799499daf6bf881321db2e4b3f7fa27c58ed66515',
'2311bf023218afe93d650cac03abb7a40f7fa55c08d2608bc82eb37db

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

['whmxw rgsxwo uqxcvnt rxbxnta', 'aveegaxr xzntqzvnxgzvr hzxctqoxnj', 'zxoyvzn wgbuhntqo', 'vooxontvoj cxqnhvr onveexzs uqxcvnt ogrhnxgzo', 'ctqexohayv ntwyzgrgsxto', 'durgfxk ogrhnxgzo', 'nhqvmxn vx vooxonvzno', 'omx rxet', 'gqmxn ogenfvqt xzw', 'xfgqp ntwyzgrgs

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

(refer top_position database results)xto']750679f']817273c']5b0ba0e']c3ebe99']423b3c6']

In [ ]:

```

```

## 5. Unsupervised Learning - Clustering

In [77]:

```
col_list = data.select_dtypes(exclude=['object']).columns.tolist()
col_list
```

Out[77]:
```
['orgyear',
 'ctc',
 'ctc_updated_year',
 'ctc_binned',
 'company_median_CTC',
 'email_avg_CTC',
 'job_pos_avg_CTC_accross_all_companies',
 'job_pos_avg_CTC_accross_same_company',
 'distinct_job_pos_per_email',
 'ctc_updated_cnt_same_company',
 'years_of_experience',
 'avg_ctc_by_company_dept_exp',
 'median_ctc_by_company_dept_exp',
 'max_ctc_by_company_dept_exp',
 'min_ctc_by_company_dept_exp',
 'count_ctc_by_company_dept_exp',
 'flag_designation',
 'flag_class',
 'company_mean_ctc',
 'flag_tier']
```

## 5.1 Label encoding/ One- hot encoding

In [78]:
```
# Label Endoing already done .
# We have already created Object data type columns for company_hask ,email_hash and job_position like company_median_CTC , ema
# job_pos_avg_CTC_accross_all_companies, job_pos_avg_CTC_accross_same_company , distinct_job_pos_per_email, ctc_updated_cnt_sa
# avg_ctc_by_company_dept_exp ,median_ctc_by_company_dept_exp and flag values
```

In [79]:
```
# Dropping object /categorical columns
df = data.copy()
df.drop(['company_hash','email_hash','job_position','ctc_binned'],axis = 1,inplace =True)
```

In [80]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205809 entries, 0 to 205808
Data columns (total 19 columns):
 #   Column                               Non-Null Count   Dtype  
---  ------                               --------------   -----  
 0   orgyear                              205809 non-null  float64
 1   ctc                                  205809 non-null  int64  
 2   ctc_updated_year                     205809 non-null  int32  
 3   company_median_CTC                   205809 non-null  float64
 4   email_avg_CTC                        205809 non-null  float64
 5   job_pos_avg_CTC_accross_all_companies 205809 non-null  float64
 6   job_pos_avg_CTC_accross_same_company  205809 non-null  float64
 7   distinct_job_pos_per_email           205809 non-null  int64  
 8   ctc_updated_cnt_same_company         205809 non-null  int64  
 9   years_of_experience                  205809 non-null  float64
 10  avg_ctc_by_company_dept_exp          205809 non-null  float64
 11  median_ctc_by_company_dept_exp       205809 non-null  float64
 12  max_ctc_by_company_dept_exp          205809 non-null  int64  
 13  min_ctc_by_company_dept_exp          205809 non-null  int64  
 14  count_ctc_by_company_dept_exp        205809 non-null  int64  
 15  flag_designation                     205809 non-null  int64  
 16  flag_class                           205809 non-null  int64  
 17  company_mean_ctc                     205809 non-null  float64
 18  flag_tier                            205809 non-null  int64  
dtypes: float64(9), int32(1), int64(9)
memory usage: 29.0 MB
```

## 5.2 Standardization of data

In [81]:

```
# Standardising Values
std_scaler = StandardScaler()
std_scaler.fit(df)
std_df = std_scaler.transform(df)

std_df = pd.DataFrame(std_df,columns=df.columns)
std_df.head()
```

Out[81]:

| | orgyear | ctc | ctc_updated_year | company_median_CTC | email_avg_CTC | job_pos_avg_CTC_accross_all_companies | job_pos_avg_CTC_a |
|---|---|---|---|---|---|---|---|
| **0** | 0.017585 | -0.099295 | 0.280511 | -0.056315 | -0.100081 | 1.525129 | |
| **1** | 0.049050 | -0.154371 | -0.474101 | -0.081104 | -0.155593 | -0.389329 | |
| **2** | 0.001852 | -0.023036 | 0.280511 | 0.079297 | -0.042434 | -0.269257 | |
| **3** | 0.033317 | -0.133188 | -0.474101 | -0.008195 | -0.134242 | -0.269257 | |
| **4** | 0.033317 | -0.073875 | -0.474101 | -0.088395 | -0.074460 | -0.389329 | |

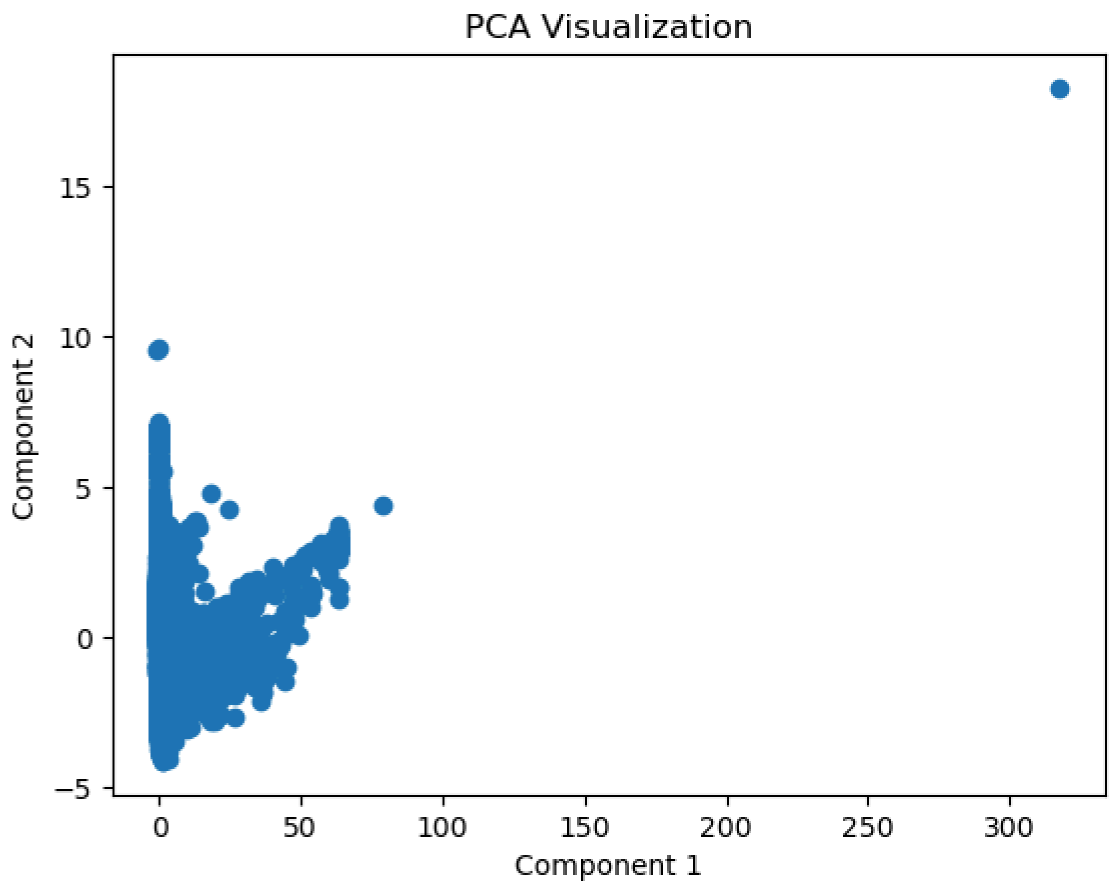
In [82]:

```
std_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205809 entries, 0 to 205808
Data columns (total 19 columns):
 #   Column                               Non-Null Count   Dtype  
---  ------                               --------------   -----  
 0   orgyear                              205809 non-null  float64
 1   ctc                                  205809 non-null  float64
 2   ctc_updated_year                     205809 non-null  float64
 3   company_median_CTC                   205809 non-null  float64
 4   email_avg_CTC                        205809 non-null  float64
 5   job_pos_avg_CTC_accross_all_companies 205809 non-null  float64
 6   job_pos_avg_CTC_accross_same_company  205809 non-null  float64
 7   distinct_job_pos_per_email           205809 non-null  float64
 8   ctc_updated_cnt_same_company         205809 non-null  float64
 9   years_of_experience                  205809 non-null  float64
 10  avg_ctc_by_company_dept_exp          205809 non-null  float64
 11  median_ctc_by_company_dept_exp       205809 non-null  float64
 12  max_ctc_by_company_dept_exp          205809 non-null  float64
 13  min_ctc_by_company_dept_exp          205809 non-null  float64
 14  count_ctc_by_company_dept_exp        205809 non-null  float64
 15  flag_designation                     205809 non-null  float64
 16  flag_class                           205809 non-null  float64
 17  company_mean_ctc                     205809 non-null  float64
 18  flag_tier                            205809 non-null  float64
dtypes: float64(19)
memory usage: 29.8 MB
```

## 5.3 PCA Visualizing the data in 2D

In [83]:
```
# PCA PLot
pca = PCA(n_components = 2)
components = pca.fit_transform(std_df)

plt.scatter(components[:,0],components[:,1])
plt.title('PCA Visualization')
plt.xlabel("Component 1")
plt.ylabel("Component 2")
plt.show()
```




## 5.4 Optimal value of K (no of clusters)

- Using Elbow Method
- Using silhouette score

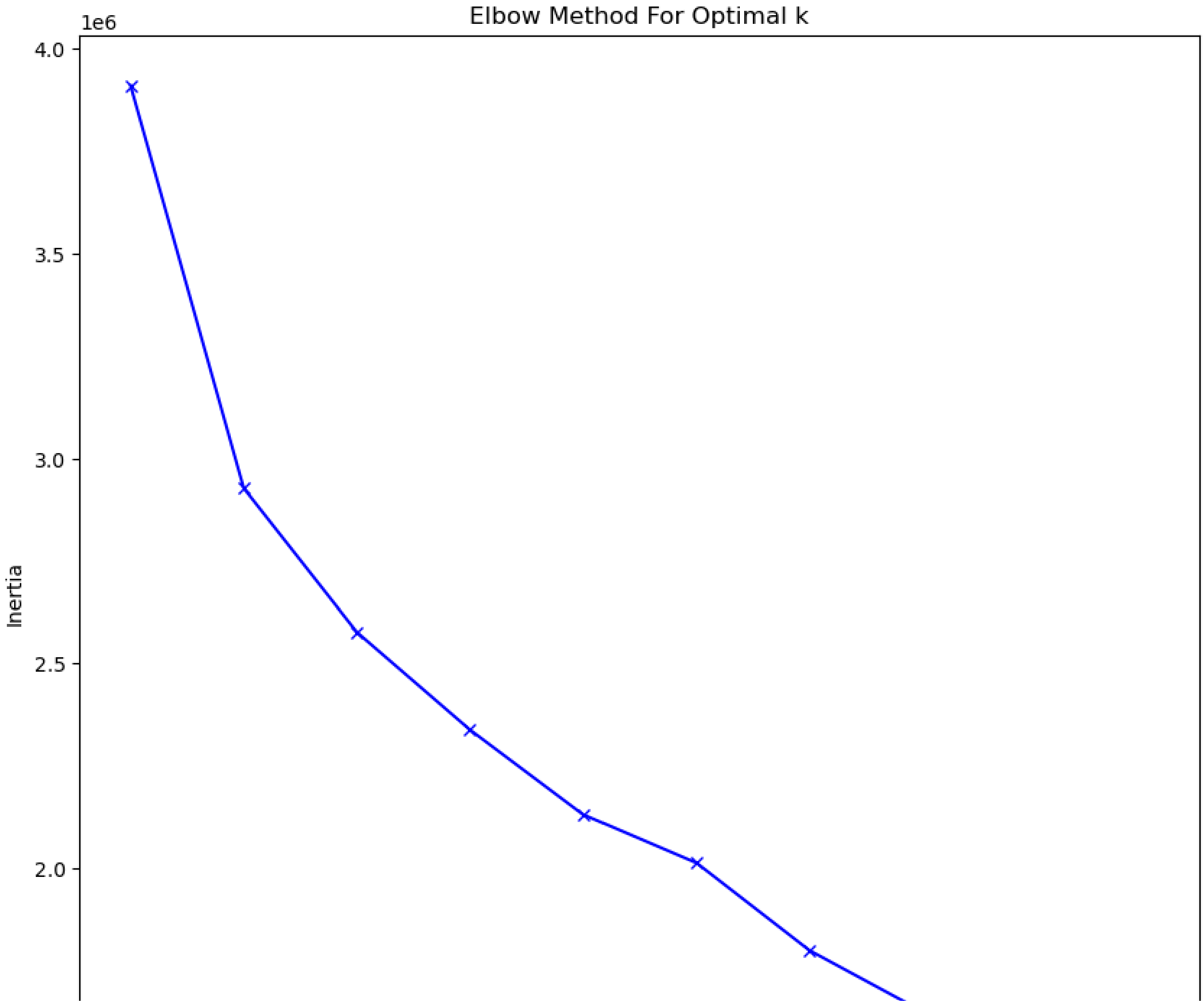
In [84]:
```
# Elbow Method
inertia = []
K = range(1, 11)
for k in K:
    kmeans = KMeans(n_clusters=k, random_state=10)
```

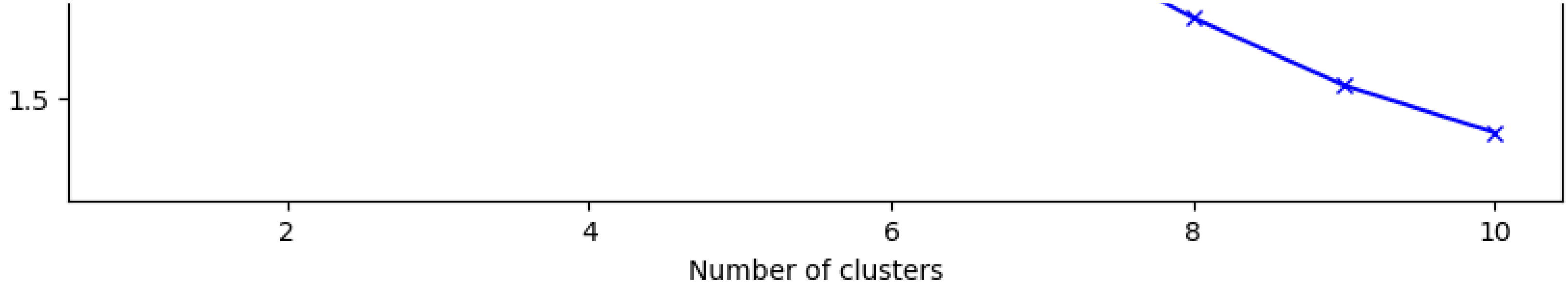
```
    kmeans.fit(std_df)
    inertia.append(kmeans.inertia_)
```

```
E:\Anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
E:\Anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
E:\Anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
E:\Anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
E:\Anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
E:\Anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
E:\Anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
E:\Anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
E:\Anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
E:\Anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

In [85]:

```
plt.figure(figsize=(10, 10))
plt.plot(K, inertia, 'bx-')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method For Optimal k')
plt.show()
```


Elbow Method For Optimal k
1e6
4.0
3.5
3.0
2.5
2.0
Inertia




## 5.5 Clustering Algorithms

- K Means Clustering (K Means++)
- Hierarchial Clustering

#### K Means Clustering

In [86]:
```
# K Means Clustering
kmeans_model = KMeans(n_clusters=2, random_state=11)
kmeans_model.fit(std_df)
cluster_labels = kmeans_model.labels_


print(kmeans_model.inertia_)
```

```
E:\Anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
2926825.1045502154
```

In [87]:
```
kmeans_model.predict(std_df)
```

Out[87]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [88]:
```
check = pd.DataFrame(kmeans_model.predict(std_df))
check.value_counts()
```

Out[88]:
```
0    205022
1       787
Name: count, dtype: int64
```

### Hierarchial Clustering

In [89]:
```
# Hierarchial Clustering

# Taking sample of dataset as data set count very high to be able to run in Hierachila Clsutering with low specs.
sample_std_df = std_df.sample(20000)

hierarchy_model = AgglomerativeClustering(n_clusters = 2, metric = 'euclidean', linkage= 'ward')
hierarchy_model.fit_predict(sample_std_df)
```

Out[89]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [90]:
```
pd.DataFrame(hierarchy_model.fit_predict(sample_std_df)).value_counts()
```

Out[90]:
```
0    19999
1        1
Name: count, dtype: int64
```

## Observation:

- Plotiing the 2D visulization using PCA
- Using Elbow method found optimal value of K (no. of cluster) to be 2
- WCSS value / intertia of K Means model is 2926825.10

In [93]:
```
top_20_jobs_data['job_position'].values
```

Out[93]:
```
array(['other', 'fullstack engineer', 'backend engineer', ..., '', '', ''],
      dtype=object)
```

# 6. Actionable Insights & Recommendations

## 6.1 EDA Analysis

- There are 205843 records and 7 columns in dataset
- `Unnamed: 0` column only represent Index of dataset / record number. Dropping the column as it will not impact the clustering.
- 34 duplicated records found , dropped them. Now the shape of dataset is 205809 rows × 6 columns
- Columns `company_hash` , `orgyear` and `job_position` have null values present in it. Around 25.5% of records in `job_position` are nulls/missing.
- Converted data type of `ctc_updated_year` to int as it represent Year and has all valid values.
- `orgyear` has 86 (cosidering 2025 orgyear also as invlaid) invalid records OR 73 (cosidering 2025 orgyear as vlaid) invalid records in it (besides NaN values). It has invalid values like like 2031 , 2106 , 0 , 20165 , etc. which is not possible considering it Employment start date (year) .
- The `email_hash` - bbace3cc586400bbc65765bc6a16b77d8913836cfc98b77c05488f02f5714a4b is most occuring out of all other. It appears 10 times which means
- Top 10 valid (non nan) job positions based on record count are 'backend engineer', 'fullstack engineer', 'other', 'frontend engineer', 'engineering leadership', 'qa engineer', 'data scientist','android engineer', 'sdet' and 'devops engineer'. Together top 10 job positions make up 64% of dataset.
- Count of outliers is less in all 3 numeric columns. Outlier percentage in `ctc` column is 6.3% , `orgyear` is 3.7% and in `ctc_updated_year` is 1.4%.
- Binned ctc column and created a new column `ctc_binned` and found that around 30% of records have current ctc in range 500K-1000K , 27% of records have ctc in range 1500K-10000K and 22% of records have ctc in range 100K-500K. Very few(around 1%) records have ctc in range 10000K-15000K or more.

## 6.2 Feature Engineering

### Created new numeric datatype columns by encoding object columns. We can then use that in clustering (instead of their object column counterpart)

- Grouping by company_hash and calculating the median Cost to Company (CTC) of its employees can be a useful encoding approach for clustering.
- Group by email_hash and calculating the average CTC for each unique email_hash can be another a useful encoding approach for clustering.
- Group by job_position and calculating the average CTC for each unique job_position can be another a useful encoding approach for clustering.

- Group by company_hash and job_position and calculating the average CTC for each unique company_hash and job_position combination can be another a useful encoding approach for clustering.
- Group by email_hash and finding number of unique job_position can be another a useful encoding approach for clustering.
- Group by company_hash and email_hash then finding number of unique orgyear can we a way to find how many times a person salary got incremented/ they got promoted in same company.
- Years of Experience column created using orgyear and current date.

## 6.3 Univariate and Bivariate Analysis¶

- Plotted plots for top_20_jobs_data , top_20_email_data ,top_20_company_data
- Heat map showcasing correlation was plotted.
- Box Plot were plotted showing outlier distribution in orgyear,

ctcand ctc_updated_ye

## 6.4 Manual Clustering

- Manual clustering was done and new columns (flag_class, flag_designation and flag_tier) were created based on the rules and answered some of the important analysis questions metioned below:

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

['5b4bed51797140db4ed52018a979db1e34cee49e27b4885c3fdfacea9f8144f6',
'0f0d1bf4233dadef653775c3f981f0ccde1dc20df36c432e934262407f79a7d6',
'7683974378d0f5bacf95632f130a60cfa3ca39e368eec9b0d50d7719c193d758',
'0f0e7a9db34d1317498d9378a1bd0150bfa022ac6ba3a02feaa740b2ae6f9927',
'a35a5abbe9fb056421bdd9aca4440acfb93e37c823564d952a50c40583920598',
'76708a11cb61a030ff3da827b0fd19aff536c3793c181689535e04c729dcfddc',
'68aa38470922a03f6022280b2a13c6f5ab6a717f70c77a0f875262a742947ce3',
'2f9a4241053f76b2f8c50ea593a90586d38b3f0e08c141d56edad2b811c0829c',
'3c85c094eaeb923add569ed91b8fbd6f8d8e8194dbaff5614338dbafb4c88074',
'1f8e216b2328e7764f79dbd66deaf008b409870ae992fe6eba3f914c6423b3c6']

### Top 10 employees of data science in each company earning more than their peers - Class 1

['bd222ea783ee372da4e0ad60fdccec0b8f37999a032025d8a83d9864bdb975ec',
'268a5aa92f0b6d0c675fc9cc1e300eb0c5930a3a139a23bc924d036bc6f3e3a5',
'cda8d723438e81185d2ee8c348870a4612eea974cdb2dbde0e99895d201a88ee',
'6b6dd66bae787dd4dd417e1777f8ea5a057257e9019995056ab8ad566995aca2',
'6ad86d120e39db485331f9a0b2b1f15ce2a7bdaee778ab0e1afea8f486d84eb8',
'4ddef8762b7585c6ee7b8c06834778f3aa00eb3be312b05e567e74969695264c',
'75f5b46d47310c3923e93329a62a1aa78d478803f0a685fec70995f380c0f5bc',
'544e75b477f8644eb71281133c62c19732547837e80e51060928888e0e7f4e90',
'15adaeb2eef9c0ee8a0f18e189bf426be390f5d1e911fdb086aa4e2b567b35ab',
'3c64901d83458f3b7b8eed6fb529ee3a4c14d49339c39810aa751621cc3ebe99']

### Bottom 10 employees of data science in each company earning less than their peers - Class 3

['690f6fdab1ab7514a6a9325ebd6cfe910dbf12d46b6fdef54f0e06c098a66dcd',
'8001bc017fbe95541d23f5780c3edb988b7d9b2225e39eec4629758f1257d2fe',
'bd9c04a574090e05b366a81cdb2f3f565d0c60fa8b1647949a3cc44ad282c485',
'e374eea75640881206a21894f69190138c2c0535277dc1909cec80f734e9a08f',
'ab2dc9db23c3104f0b6b3dbd4cdd5bfb9e5829b8b7943d5185fb535fe7c6b7c1',
'3175d03fd4618eb293d6f5a1d13d42a0c79f68e9acaaa31e0f4da21ae8f1b6f0',
'3675f79c7e05de96ccf189c818b84b487cb1aa3f6b80e886ba5cd94a5d1e79e8',
'fb64af615420e06d46a1965f59068b34460fb3cbe70541c6c1852e8458396a62',
'8274b3188470cd1c4914e7face490111e27f239457e62d7edb2ee1353e8cb18b',
'3cc0c85d198d0e56a4cdefb6496333f59b97f87c2932629c2de53d4cd5b0ba0e']

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

['3505b02549ebe2c95840ac6f0a35561a3b4cbe4b79cdb15d794a6ac066b66644',
'f2b58aeed3c074652de2cfd3c0717a5d21d6fbcf342a786928c5fd38c860fa45',
'23ad96d6b6f1ecf554a52f6e9b61677c7d73d8a409a143cee85eda1557f516c8',
'b8a0bb340583936b5a7923947e9aec21add5ebc50cd60bf6953ea67074932d41',
'f7e5e788676100d7c4146740ada9e2f8974defc01f571d34a3ff1ccf6b68c0a8',

'9af3dca6c9d705d8d42585ccfce2627f00e1629130d14ef814d03bd2ac256596',
'80ba0259f9f59034c4927cf3bd38dc9ce2eb60ff18135bf9012feac4c1169c23',
'b995d7a2ae5c6f8497762ce04dc5c04ad6ec734d70802a94f83e17431884e907',
'f0f2005505c707dbdd2c86ca1587c26f822a004e86a8ec9caebc6145336fe83d',
'afa111053d280ef49ea791a3b5f7d171f961f4bd8ec7243613aa944f3817273c']

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

['5b4bed51797140db4ed52018a979db1e34cee49e27b4885c3fdfacea9f8144f6',
'1b95e7ba0ee82100ca5a034239fa0203a1bec14280b82a5cc81a192bb7c82df3',
'1b95e7ba0ee82100ca5a034239fa0203a1bec14280b82a5cc81a192bb7c82df3',
'76fec55391d520a956adabf982d88faf6842b6188b901feddff30a2a307d0c66',
'38e99d23328a60acdf25e8c4fa7616661093fd4874151a9a58ea0f0650e02823',
'9e785d33821db67c01becc1c36f901d79d3142c1d13bd84cf8aa670f30717e20',
'a071c4cd6d423e8d1841ba6133e6c4684f4eaba7dc15261313f206d04e640ebd',
'634fd283565b8954513a6ad0e47cedb0fa8847923149fbae5f3d3889075837b6',
'89f343bf01094accb8b0b2c799499daf6bf881321db2e4b3f7fa27c58ed66515',
'2311bf023218afe93d650cac03abb7a40f7fa55c08d2608bc82eb37db750679f']

## Top 10 companies (based on their CTC)

['whmxw rgsxwo uqxcvnt rxbxnta', 'aveegaxr xzntqzvnxgzvr hzxctqoxnj', 'zxoyvzn wgbuhntqo', 'vooxontvoj cxqnhvr onveexzs uqxcvnt ogrhnxgzo', 'ctqexohayv ntwyzgrgsxto', 'durgfxk ogrhnxgzo', 'nhqvmxn vx vooxonvzno', 'omx rxet', 'gqmxn ogenfvqt xzw', 'xfgqp ntwyzgrgsxto']

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

(refer top_position database results)

## 6.5 Unsupervised Learning - Clustering

- After doing encoding and Standardization of data optimal value of K (no. of clusetrs) was found using Elbow method.

- Clustering algorithms used were Kmeans and Hierarchial clustering
- Plotiing the 2D visulization using PCA
- Using Elbow method found optimal value of K (no. of cluster) to be 2
- WCSS value / intertia of K Means model is 2926825.1092

# Actionable insided and recommendations

## Actionable Insights

1. Top Performers:

- Tier 1 Employees: These employees are top earners within their companies. They could be potential candidates for leadership roles or key projects.

- Class 1 Data Science Employees: These individuals are top earners within their data science roles across different companies. They can be further motivated or recognized for their exceptional performance.

2. Low Performers:

- Tier 3 Employees: These employees are at the lower end of the earnings spectrum within their companies. They might need additional training, support, or opportunities for growth.

- Class 3 Data Science Employees: These individuals are earning less than their peers in the data science domain. Consider conducting performance reviews or providing additional resources to help them improve.

3. Job Position Analysis:

- Most Job Positions are hiring and developing talent in the top 10 job positions like backend engineer, fullstack engineer, and frontend engineer. These roles are critical and make up a significant portion of the workforce.

- Job Position Distribution: Keep an eye on the distribution of job positions and ensure a balanced allocation of resources and opportunities across different roles.

4. Company Analysis:

- Top Companies: The top 10 companies with the highest CTC values are attractive employers. Investigate their practices and strategies to replicate their success in other companies.

- Company Clustering: Grouping similar companies based on CTC and other factors can help tailor specific strategies for different clusters of companies.

5. Experience-Based Insights:

- Years of Experience: Employees with 5-7 years of experience and higher earnings can be considered for mentorship roles or leadership tracks.

## Recommendations

1. Employee Development Programs:

- Implement targeted development programs for Tier 3 and Class 3 and Designation 3 employees to help them improve their skills and performance.
- Offer mentorship and leadership training for Tier 1 employees to prepare them for higher responsibilities.

2. Recognition and Rewards:

- Recognize and reward top performers (both Tier 1 and Class 1) to keep them motivated and engaged.
- Consider implementing incentive programs to boost performance across all tiers/class/designation.

3. Talent Acquisition and Retention:

- Focus on hiring and retaining talent in the most common and critical job positions.
- Use the insights from clustering to create tailored retention strategies for different clusters of companies.

4. Performance Reviews and Feedback:

- Conduct regular performance reviews and provide constructive feedback to low performers.
- Use data-driven insights to identify areas for improvement and provide necessary resources.

5. Data-Driven Decision Making:

- Leverage the insights from clustering to make informed decisions about resource allocation, training programs, and employee engagement initiatives.
- Continuously monitor and update the clustering models to ensure they reflect the latest trends and patterns.

In [ ]: