scaler-clustering

November 29, 2024

0.1 About Scaler

Scaler is an online tech-versity offering intensive computer science & Data Science courses through live classes delivered by tech leaders and subject matter experts. The meticulously structured program enhances the skills of software professionals by offering a modern curriculum with exposure to the latest technologies. It is a product by InterviewBit.

0.1.1 Business Problem

To cluster a segment of learners from the Scaler database based on their job profile, company, and other relevant features, with the goal of profiling the best companies and job positions to work for. The resulting clusters should group learners with similar characteristics.

0.1.2 Dataset

Column Name	Description
`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)

Importing Required Libraries

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

from scipy import stats
from sklearn.impute import KNNImputer
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

```
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster

from matplotlib import pyplot as plt
import seaborn as sns

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

[2]:
```
palette = ['#0000ff', '#ffffff', '#000000', '#99c2ff']
```

Read Dataset

[3]:
```
df = pd.read_csv(r'scaler_clustering.csv', index_col=0)
df.head()
```

[3]:
```
                  company_hash  \
0               atrgxnnt xzaxv
1  qtrxvzwt xzegwgbb rxbxnta
2                ojzwnvwnxw vx
3                    ngpgutaxv
4                   qxen sqghu

                                          email_hash  orgyear      ctc  \
0  6de0a4417d18ab14334c3f43397fc13b30c35149d70c05…   2016.0  1100000
1  b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10…   2018.0   449999
2  4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9…   2015.0  2000000
3  effdede7a2e7c2af664c8a31d9346385016128d66bbc58…   2017.0   700000
4  6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520…   2017.0  1400000

        job_position  ctc_updated_year
0              Other            2020.0
1  FullStack Engineer            2019.0
2    Backend Engineer            2020.0
3    Backend Engineer            2019.0
4  FullStack Engineer            2019.0
```

[4]:
```
print("Shape of the data: ", df.shape)
print("The Given Dataset has {} rows and {} columns".format(df.shape[0], df.
 ↪shape[1]))
print("Columns: ", df.columns.to_list())
```

```
Shape of the data:  (205843, 6)
The Given Dataset has 205843 rows and 6 columns
Columns:  ['company_hash', 'email_hash', 'orgyear', 'ctc', 'job_position',
'ctc_updated_year']
```

0.1.3 Shape

- The dataset comprises 205843 rows and 6 columns, representing a volume of data.
- Each row corresponds to information about the learners details.

0.1.4 Data Structure

[5]:
```
df.describe()
```

[5]:

	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

[6]:
```
df.describe(include='object')
```

[6]:

```
                    company_hash  \
count                     205799
unique                     37299
top     nvnv wgzohrnvzwj otqcxwto
freq                        8337

                                               email_hash      job_position
count                                              205843            153279
unique                                             153443              1016
top     bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7…  Backend Engineer
freq                                                   10             43554
```

[7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 205843 entries, 0 to 206922
Data columns (total 6 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   company_hash      205799 non-null  object
 1   email_hash        205843 non-null  object
 2   orgyear           205757 non-null  float64
 3   ctc               205843 non-null  int64
 4   job_position      153279 non-null  object
 5   ctc_updated_year  205843 non-null  float64
dtypes: float64(2), int64(1), object(3)
```

```
memory usage: 11.0+ MB
```

[8]:
```
df.isnull().sum()
```

[8]:
```
company_hash           44
email_hash              0
orgyear                86
ctc                     0
job_position        52564
ctc_updated_year        0
dtype: int64
```

[9]:
```
print(df.duplicated().sum())
df.drop_duplicates(inplace=True)
```

```
34
```

0.1.5 Dataset Information:

- **Data Consistency**: job_position column has more number of missing values, columns like orgyear, company_hash also has missing values.
- **Data Types**: Columns are classified into integer, float and object types.
- **Data Duplicates**: There are 132 rows in the record are duplicated.

Initial Cleanup

[10]:
```
df['job_position'] = df['job_position'].apply(lambda x: re.sub('[^A-Za-z0-9␣
↪]+', '', x.lower()) if isinstance(x, str) else x)
df['company_hash'] = df['company_hash'].apply(lambda x: re.sub('[^A-Za-z0-9␣
↪]+', '', x.lower()) if isinstance(x, str) else x)
df['email_hash'] = df['email_hash'].apply(lambda x: re.sub('[^A-Za-z0-9 ]+',␣
↪'', x.lower()) if isinstance(x, str) else x)
```

[11]:
```
df['orgyear'] = df.groupby(['company_hash'])['orgyear'].transform(lambda x: x.
↪ffill().bfill())
print(df['orgyear'].isnull().sum())
df['orgyear'] = df['orgyear'].fillna(df['orgyear'].mode()[0])
```

```
70
```

[12]:
```
# Convert the data types
df['job_position'] = df['job_position'].astype('category')
df['company_hash'] = df['company_hash'].astype('category')
df['email_hash'] = df['email_hash'].astype('category')

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

0.1.6 Data Observation

[13]:
```
df.shape
```

[13]:
```
(205809, 6)
```

[14]:
```
#### Frequency of Email Id and Company Hash
# df.groupby(['email_hash','company_hash']).size().reset_index(name='count').
 ↪sort_values('count', ascending=False).head()
df[['email_hash', 'company_hash']].value_counts().reset_index(name='count').
 ↪sort_values('count', ascending=False).head()
```

[14]:
```
                                          email_hash  \
0  bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7…
2  298528ce3160cc761e4dc37a07337ee2e0589df251d736…
3  6842660273f70e9aa239026ba33bfe82275d6ab0d20124…
1  3e5e49daa5527a6d5a33599b238bf9bf31e85b9efa9a94…
4  c0eb129061675da412b0deb15871dd06ef0d7cd86eb5f7…

                company_hash  count
0  oxej ntwyzgrgsxto rxbxnta     10
2              cvrhtbgbtznhb      9
3                   ihvrwgbb      9
1        wgcxvb ntwyzgrgsxto      9
4           nyt a t oyvf sqghu      8
```

[15]:
```
df[df['email_hash'].str.
 ↪contains('bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7')]
```

[15]:
```
                     company_hash  \
24129   oxej ntwyzgrgsxto rxbxnta
46038   oxej ntwyzgrgsxto rxbxnta
72415   oxej ntwyzgrgsxto rxbxnta
103145  oxej ntwyzgrgsxto rxbxnta
118076  oxej ntwyzgrgsxto rxbxnta
121825  oxej ntwyzgrgsxto rxbxnta
124840  oxej ntwyzgrgsxto rxbxnta
145021  oxej ntwyzgrgsxto rxbxnta
153402  oxej ntwyzgrgsxto rxbxnta
160472  oxej ntwyzgrgsxto rxbxnta

                                               email_hash  orgyear     ctc  \
24129   bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7…     2018  720000
46038   bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7…     2018  720000
72415   bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7…     2018  720000
103145  bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7…     2018  720000
```

```
118076  bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7…     2018  720000
121825  bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7…     2018  660000
124840  bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7…     2018  660000
145021  bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7…     2018  660000
153402  bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7…     2018  660000
160472  bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7…     2018  660000

              job_position  ctc_updated_year
24129                  NaN              2020
46038     support engineer              2020
72415                other              2020
103145  fullstack engineer              2020
118076        data analyst              2020
121825               other              2019
124840    support engineer              2019
145021  fullstack engineer              2019
153402     devops engineer              2019
160472                 NaN              2019
```

[16]:
```
df.
↪sort_values(by=['email_hash','company_hash','orgyear','ctc','ctc_updated_year','job_positio
↪ascending=True, inplace=True)
df['job_position'] = df.groupby(['email_hash', 'company_hash', 'ctc',␣
↪'ctc_updated_year'])['job_position'].transform(lambda x: x.ffill().bfill())
```

[17]:
```
print(df.duplicated().sum())
df.drop_duplicates(inplace=True)
```

```
27308
```

0.1.7 Duplicate value check:

- **Frequency of Company/Email Hash**: It is observed that the for a individual learner there are multiple entires in the dataset with different job position name.
- **Data Types**: Columns are classified into integer, float and object types. DateTime columns are stored as Objects types.

[18]:
```
print("No. of Unique Job Positions: ", df['job_position'].nunique())
print("No. of Null in Job Positions: ",df['job_position'].isnull().sum())
```

```
No. of Unique Job Positions:  931
No. of Null in Job Positions:  25332
```

[19]:
```
## Manual clean up of job positions

# jobs_df = df['job_position'].value_counts().to_frame()
# jobs_df.reset_index(inplace=True)
# jobs_df.columns = ['job_position', 'count']
```

```python
# jobs_df.to_csv('job_positions.csv', index=False)

jobs_df = pd.read_csv('job_positions.csv')
jobs_df.head()
```

[19]:

	job_position	count	filler
0	backend engineer	43551	Backend Engineer
1	fullstack engineer	25975	Fullstack Engineer
2	other	18070	Others
3	frontend engineer	10417	Frontend Engineer
4	engineering leadership	6870	Technical lead

[20]:
```python
job_position_dict = dict(zip(jobs_df['job_position'], jobs_df['filler']))
# job_position_dict
```

[21]:
```python
df['job_position'] = df['job_position'].map(job_position_dict)
```

[22]:
```python
print("Null count: ", df['job_position'].isnull().sum())
print("Number of Uniques: ", df['job_position'].nunique())
```

```
Null count:  25335
Number of Uniques:  41
```

[23]:
```python
# Encode the categorical column
# Temporarily fill NaN values with a placeholder
df['job_position'].fillna('missing', inplace=True)

# Encode the categorical column
le = LabelEncoder()
df['job_position_encoded'] = le.fit_transform(df['job_position'])

# Replace the placeholder back to NaN
df['job_position_encoded'].replace(np.where(le.classes_ == 'missing')[0][0], np.
↪nan, inplace=True)

# Apply KNN imputer
imputer = KNNImputer(n_neighbors=5)
df['job_position_encoded'] = imputer.fit_transform(df[['job_position_encoded']])

# Decode the imputed values back to original categories
df['job_position'] = le.inverse_transform(df['job_position_encoded'].
↪astype(int))



# Drop the encoded column
df.drop(columns=['job_position_encoded'], inplace=True)
```

[24]:
```python
df.isnull().sum()
```

[24]:
```
company_hash        39
email_hash           0
orgyear              0
ctc                  0
job_position         0
ctc_updated_year     0
dtype: int64
```

[25]:
```
print(df['company_hash'].nunique())
df.dropna(subset=['company_hash'], inplace=True)
```

```
37299
```

[26]:
```
df.isnull().sum()
```

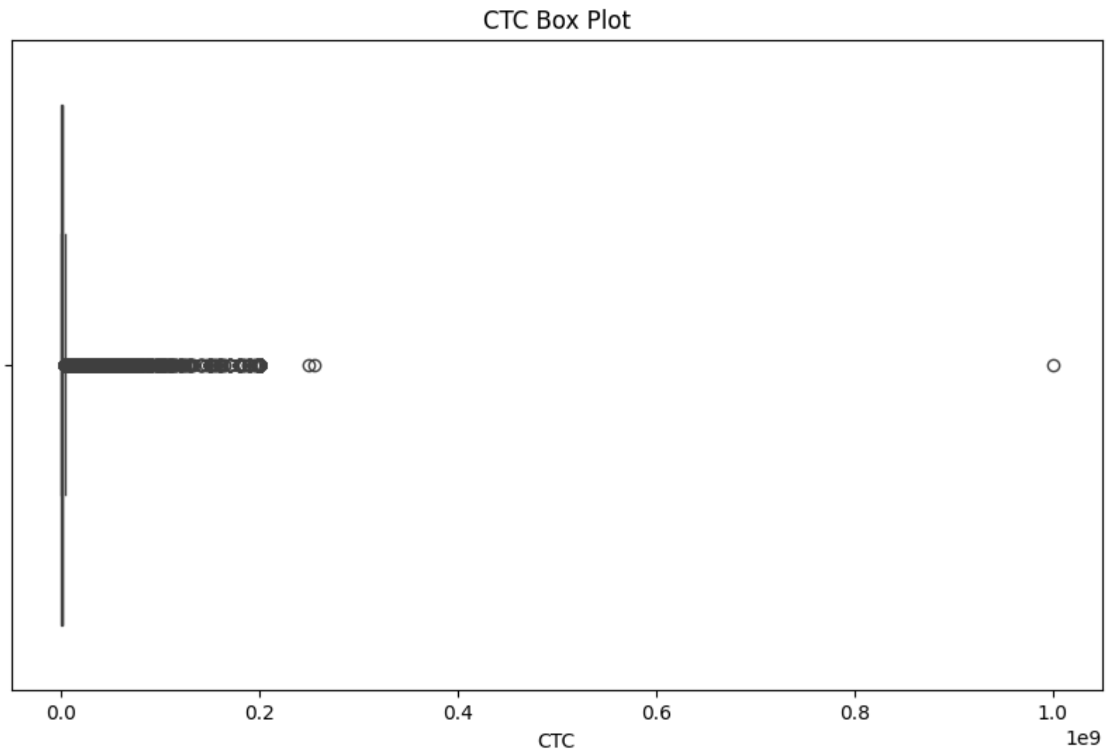
[26]:
```
company_hash        0
email_hash          0
orgyear             0
ctc                 0
job_position        0
ctc_updated_year    0
dtype: int64
```

0.1.8 Exploratory Data Analysis (EDA)

[27]:
```
# CTC Column checks
print(df['ctc'].min(), df['ctc'].max())

# Calculate Percentiles
lower_bound = df['ctc'].quantile(0.01)
upper_bound = df['ctc'].quantile(0.99)

print(f"Lower Bound: {lower_bound}, Upper Bound: {upper_bound}")
```

```
2 1000150000
Lower Bound: 35000.0, Upper Bound: 16000000.0
```

[28]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x=df['ctc'])
plt.title('CTC Box Plot')
plt.xlabel('CTC')
plt.show()
```



[29]:
```
df['CTC_zscore'] = stats.zscore(df['ctc'])

# Identify outliers (e.g., Z-score > 3 or < -3)
outliers = df[(df['CTC_zscore'] > 3) | (df['CTC_zscore'] < -3)]

print("Outliers based on Z-score:")
print("Number of outliers: ", outliers.shape[0])
print("Percentage of outliers: ", outliers.shape[0] / df.shape[0] * 100)
```

```
Outliers based on Z-score:
Number of outliers:  1499
Percentage of outliers:  0.8399547242550235
```

[30]:
```
outliers.sort_values(by='CTC_zscore', ascending=True).head()
```

[30]:
```
            company_hash                                         email_hash  \
168918      ftrro evqsg  55c7eef700e87fcd506c731be6dbbdcb79bf709678c2eb…
50457        wgzwtznqxd  c240203d659ee5ef1a7ed9d8368ec86b7fc1d21e10701b…
15755         wgszxkvzn  405acce4415219a5001f40c37ca2e5f07d433a8e769641…
77447         wgszxkvzn  405acce4415219a5001f40c37ca2e5f07d433a8e769641…
141547   zgn vuurxwvmrt  a7b47e958b5a48f375cb74e23537b601396356545a19c5…

        orgyear       ctc       job_position  ctc_updated_year  CTC_zscore
```

```
168918     2002  39600000     Technical lead              2019    3.006501
50457      2017  39800000             Others              2020    3.022664
15755      2017  39900000        QA Engineer              2020    3.030746
77447      2017  39900000   Support Engineer              2020    3.030746
141547     2021  40000000  Frontend Engineer              2019    3.038828
```

[31]:
```
outliers.sort_values(by='CTC_zscore', ascending=False).head()
```

[31]:
```
                              company_hash  \
72925         whmxw rgsxwo uqxcvnt rxbxnta
117948                        obvqnuqxdwgb
3301    aveegaxr xzntqzvnxgzvr hzxctqoxnj
9220         vayxxuv ntwyzgrgsxto ucn rna
107466         ytfrtnn uvwpvqa tzntquqxot

                                               email_hash  orgyear  \
72925   29a71dd13adf6d2d497571a565bb3096cf66cb46cd1ece…     2015
117948  5b4bed51797140db4ed52018a979db1e34cee49e27b488…     2018
3301    06d231f167701592a69cdd7d5c825a0f5b30f0347a4078…     2021
9220    ffc553f9b28005de1ba0ce2b546befbe7179bf0d44e2de…     2014
107466  c888824e687a535d1bd2486ae28d67e414b21b09cbee61…     2015

               ctc       job_position  ctc_updated_year  CTC_zscore
72925   1000150000  Frontend Engineer              2020   80.635227
117948   255555555  Frontend Engineer              2016   20.459371
3301     250000000  Frontend Engineer              2020   20.010388
9220     200000000         Co founder              2020   15.969541
107466   200000000       Data Analyst              2020   15.969541
```

[32]:
```
df = df[df['ctc']>=100000]
df = df[df['ctc']<=15000000]
```

[33]:
```
print("Median: ",df['ctc'].median())
print("Mean: ",df['ctc'].mean())
```

```
Median:  1000000.0
Mean:  1370185.5501391143
```

[34]:
```
# Create ctc into 3 categories
# df['ctc_category'] = pd.qcut(df['ctc'], q=3, labels=['low', 'medium', 'high'])
df['ctc_category'] = pd.cut(df['ctc'], bins=[0, 1000000, 5000000, 15000000],␣
 ↪labels=['low', 'medium', 'high'])
```

[35]:
```
df['ctc_category'].value_counts()
```

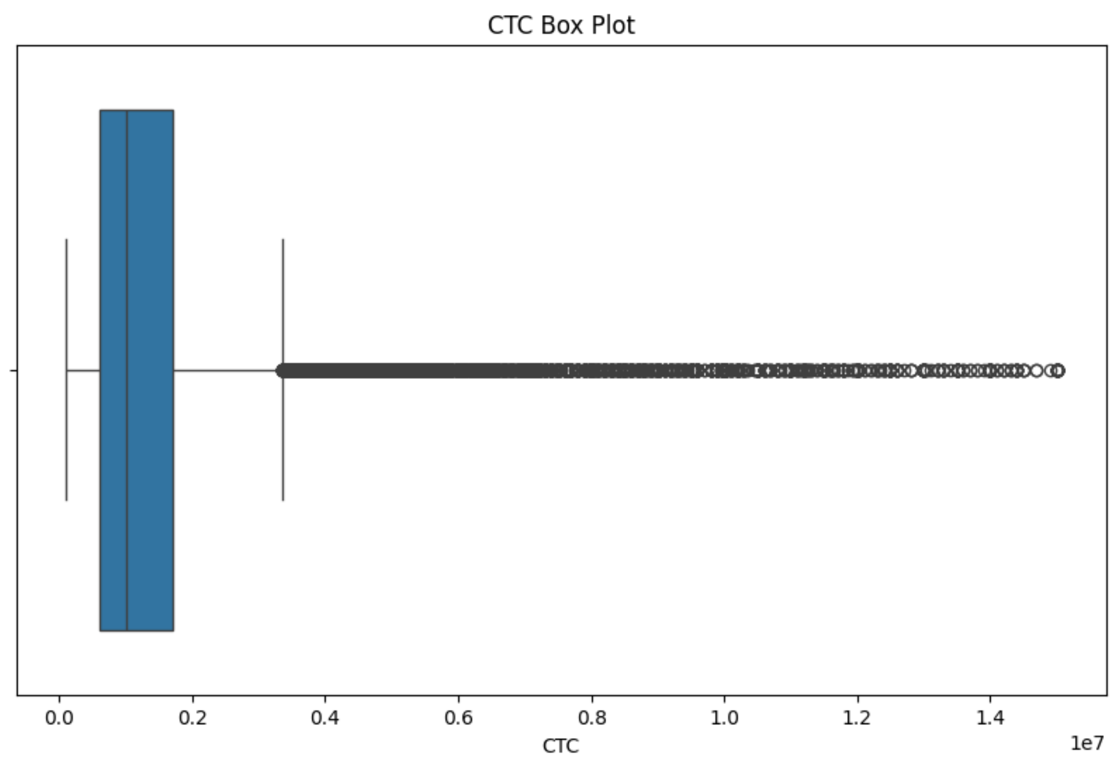
[35]:
```
ctc_category
low       91187
medium    78056
```

```
high        3277
Name: count, dtype: int64
```

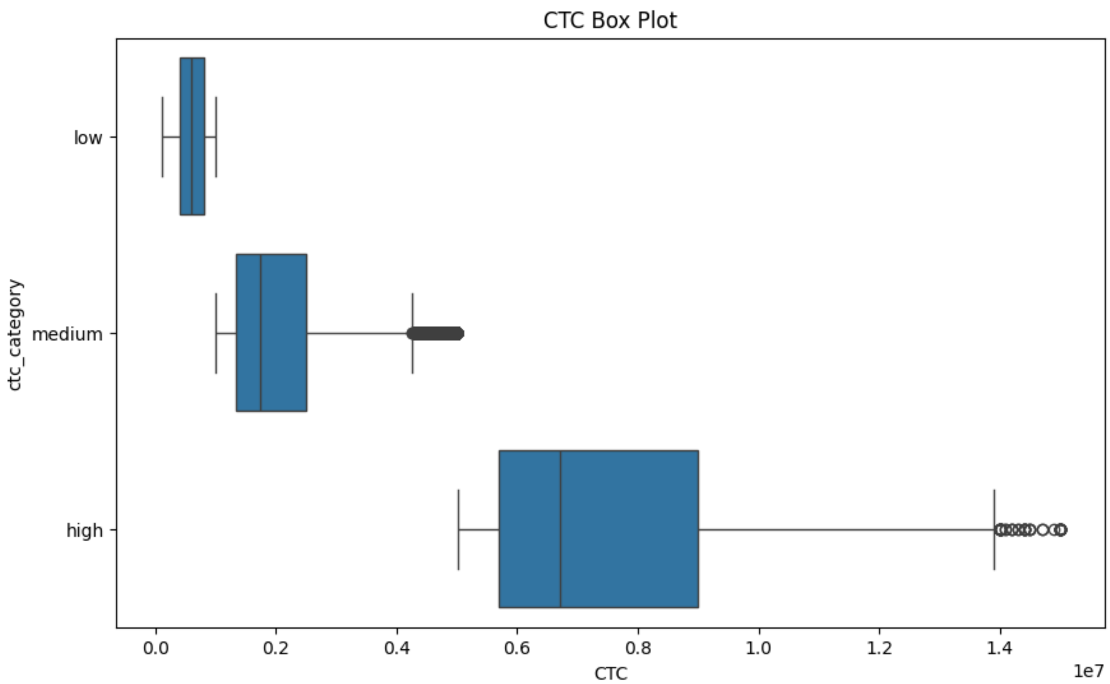
[36]:
```
df.groupby('ctc_category')['ctc'].describe()
```

[36]:
```
                 count          mean           std        min        25%  \
ctc_category
low            91187.0  5.955088e+05  2.401585e+05   100000.0   400000.0
medium         78056.0  2.017507e+06  8.878268e+05  1000008.0  1330000.0
high            3277.0  7.507861e+06  2.281781e+06  5010000.0  5700000.0

                     50%        75%         max
ctc_category
low             600000.0   800000.0   1000000.0
medium         1730000.0  2500000.0   5000000.0
high           6700000.0  9000000.0  15000000.0
```

[37]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x=df['ctc'])
plt.title('CTC Box Plot')
plt.xlabel('CTC')
plt.show()
```



[38]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x=df['ctc'], y=df['ctc_category'])
plt.title('CTC Box Plot')
plt.xlabel('CTC')
plt.show()
```



[39]:
```
print("Organisation Year Range: ", df['orgyear'].min(), df['orgyear'].max())

# Potential outliers - orgyear < 1970 people are not likely to be working after␣
↪54 years
df[(df['orgyear']<1970) | (df['orgyear']>2024)].shape
```

```
Organisation Year Range:  0 20165
```

[39]: (77, 8)

[40]:
```
df = df[(df['orgyear']>=1970) & (df['orgyear']<=2024)]
df['year_of_experience'] = 2024 - df['orgyear']
```

0.1.9 Data Pre-processing:

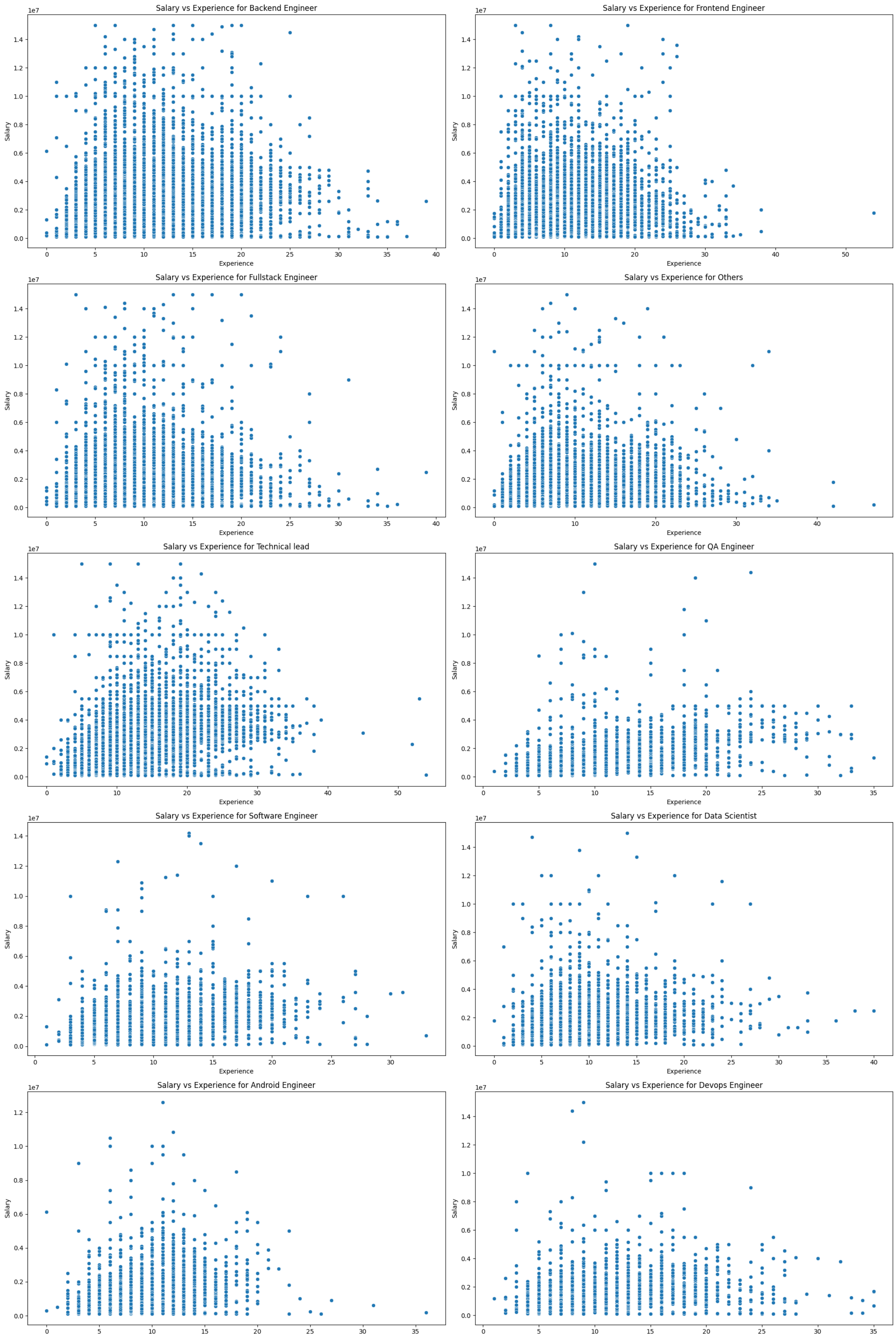
- **Job Position**: The job title are cleaned manually and mapped backed to the dataset, Missing job positions are filled using KNN Imputation
- **Company Hash**: Dropped the columns where company name not provided

- **CTC**: The dataset has beeen capping based on the domain knowledge (RS1,00,000 to 1,50,00,000)
- **Organisation Year**: The Organisation year has been capped based on the fact that person might not work after 54 years
- **Year of Experience**: The column dervied from the org year

0.1.10 Insights:

- **Distribution of ctc_category**
  - Low CTC Category: 91,187 learners
  - Medium CTC Category: 78,056 learners
  - High CTC Category: 3,277 learners
- **Skewed Distribution:**
  - The distribution of learners across the ctc_category is highly skewed, with the majority falling into the low and medium categories. Only a small fraction of learners are in the high category.
- **Outliers:**
  - The presence of outliers in the medium and high categories suggests that there are learners with significantly higher CTC values than the rest. These outliers could be due to promotions, exceptional performance, or high-paying job profiles.

[41]:
```
# Top 10 job positions, plot the salary vs experience
top_10_job_positions = df['job_position'].value_counts().head(10).index.
↪to_list()

fig, axes = plt.subplots(5, 2, figsize=(20, 30))
axes = axes.flatten()

for i, job in enumerate(top_10_job_positions):
    sns.scatterplot(data=df[df['job_position'] == job], x='year_of_experience',␣
↪y='ctc', ax=axes[i])
    axes[i].set_title(f'Salary vs Experience for {job}')
    axes[i].set_xlabel('Experience')
    axes[i].set_ylabel('Salary')

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


Salary vs Experience for Backend Engineer
Salary vs Experience for Frontend Engineer
Salary vs Experience for Fullstack Engineer
Salary vs Experience for Others
Salary vs Experience for Technical lead
Salary vs Experience for QA Engineer
Salary vs Experience for Software Engineer
Salary vs Experience for Data Scientist
Salary vs Experience for Android Engineer
Salary vs Experience for Devops Engineer
Salary
Experience
1e7

0.1.11 Insights:

- **Positive Correlation:**
  - There is a general trend where salary increases with years of experience across most job titles. However, this trend is not strictly linear and shows significant variability.
- **Plateau Effect:**
  - For many job titles, the salary seems to plateau after a certain number of years of experience. This indicates that beyond a certain point, additional years of experience do not significantly increase the salary.
- **Outliers and High Earners:**
  - High-salary outliers are present in most job titles, indicating that exceptional performance, specialized skills, or high-paying employers can lead to significantly higher salaries.
- **Is it always true that with an increase in years of experience, the CTC increases?**
  - There is no clear trend, that CTC increase based on increase in years of experience

[42]:
```
company_wise_summary_df = df.groupby(['company_hash', 'job_position',␣
↪'year_of_experience'])['ctc'].agg(['min', 'mean', 'median', 'max', 'count']).
↪reset_index()
company_wise_summary_df =␣
↪company_wise_summary_df[company_wise_summary_df['count'] > 0]
company_wise_summary_df.sample(5)
```

[42]:
```
                             company_hash        job_position  \
63643470  wvqo24 otqcxwto uqxcvnt rxbxnta    Backend Engineer
26039978                         ntqvavnv   Research Engineer
54856399               vngo ojzntr ucn rna      Data Scientist
46521244                       tdutqxnton   Frontend Engineer
57749595                            vuurt  Fullstack Engineer

          year_of_experience        min       mean     median        max  \
63643470                  16  4500000.0  4500000.0  4500000.0  4500000.0
26039978                   6   900000.0   900000.0   900000.0   900000.0
54856399                  17  1800000.0  1800000.0  1800000.0  1800000.0
46521244                   7  2200000.0  2200000.0  2200000.0  2200000.0
57749595                   8   220000.0   220000.0   220000.0   220000.0

          count
63643470      1
26039978      1
54856399      1
```

```
46521244      1
57749595      1
```

0.1.12 Feature Creation

[43]:
```
def category_mapping(ctc, mean):
  if ctc < mean:
    return 1
  elif ctc > mean:
    return 3
  else:
    return 2
```

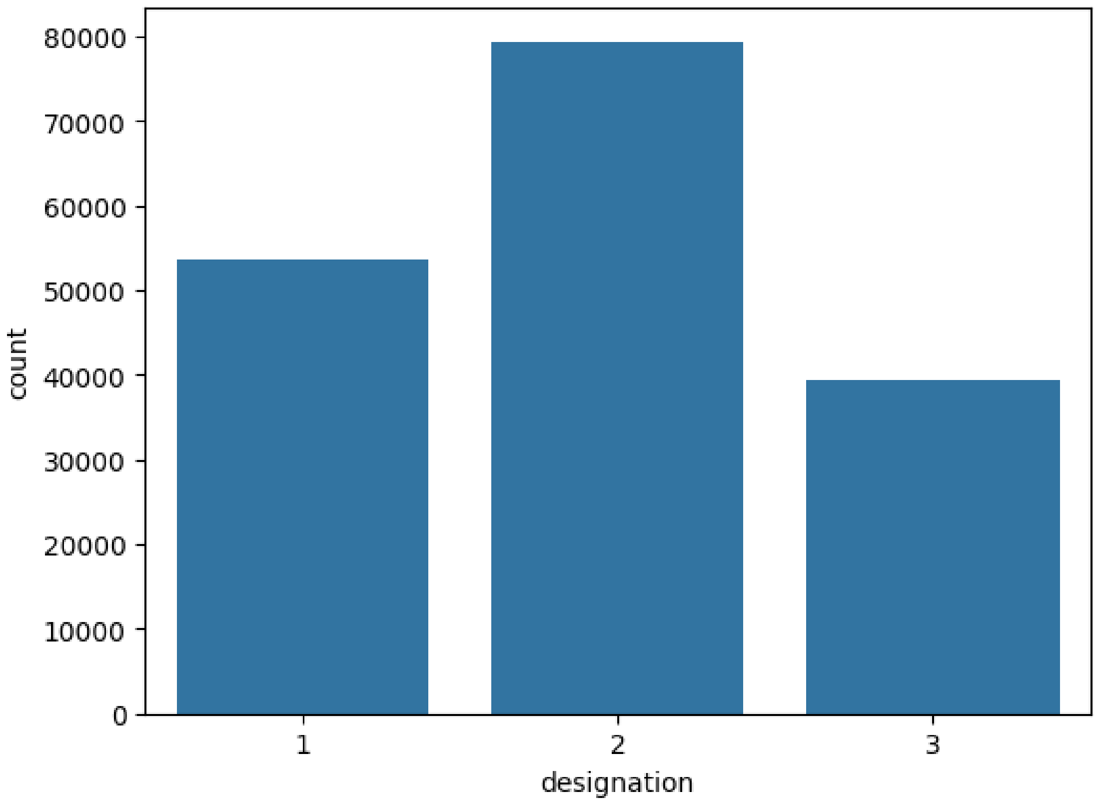
[44]:
```
df_grouped = df.groupby(['company_hash', 'job_position',␣
↪'year_of_experience'])['ctc'].mean().reset_index()
df_grouped.rename(columns={'ctc': 'com_pos_exp_ctc_mean'}, inplace=True)
df_grouped = df_grouped[df_grouped['com_pos_exp_ctc_mean'] > 0]
df_grouped.head()
```

[44]:
```
                     company_hash        job_position  year_of_experience  \
1180                            0              Others                   4
3192                         0000              Others                   7
4222                  01 ojztqsj    Android Engineer                   8
4864                  01 ojztqsj   Frontend Engineer                  13
6375  05mz exzytvrny uqxcvnt rxbxnta    Backend Engineer                   5

      com_pos_exp_ctc_mean
1180              100000.0
3192              300000.0
4222              270000.0
4864              830000.0
6375             1100000.0
```

[45]:
```
df = pd.merge(df, df_grouped, on=['company_hash', 'job_position',␣
↪'year_of_experience'], how='left')
df['designation'] = df.apply(lambda x: category_mapping(x['ctc'],␣
↪x['com_pos_exp_ctc_mean']), axis=1)
df.drop(columns=['com_pos_exp_ctc_mean'], inplace=True)
```

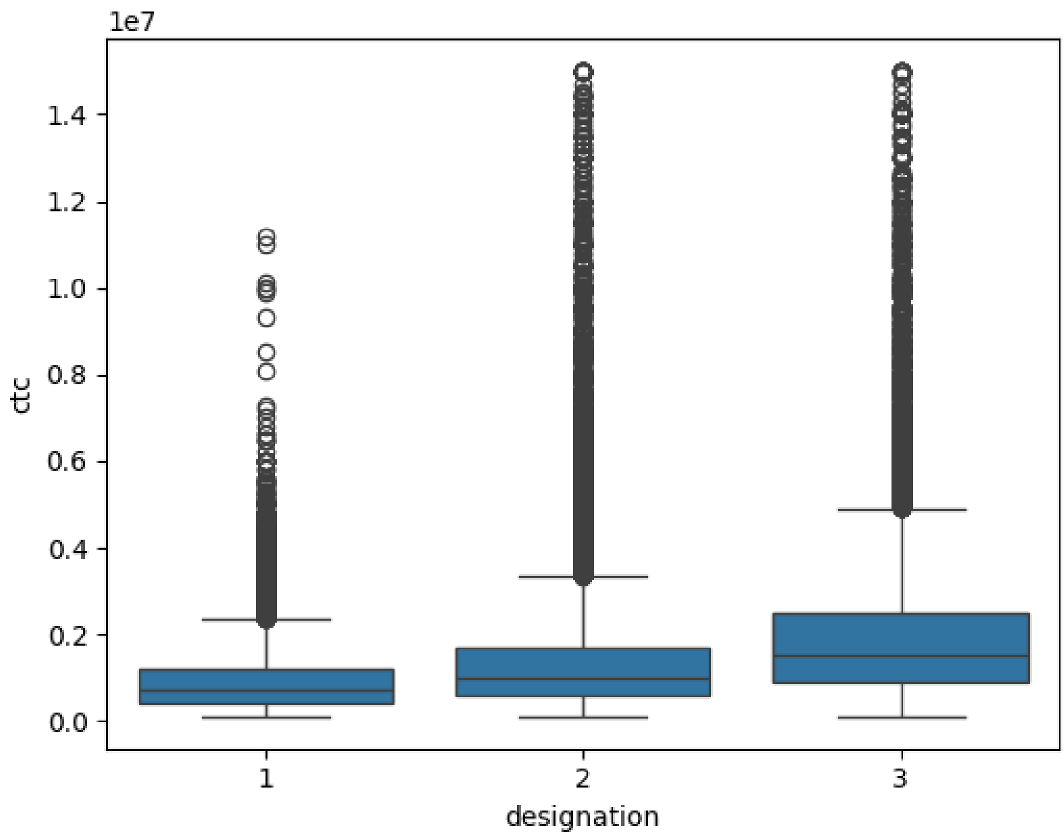
[46]:
```
sns.countplot(x='designation', data=df)
```

[46]: `<Axes: xlabel='designation', ylabel='count'>`



[47]:
```
sns.boxplot(x='designation', y='ctc', data=df)
```

[47]: <Axes: xlabel='designation', ylabel='ctc'>



[48]:
```
df_grouped = df.groupby(['company_hash', 'job_position'])['ctc'].mean().
    ↪reset_index()
df_grouped.rename(columns={'ctc': 'com_pos_ctc_mean'}, inplace=True)
df_grouped = df_grouped[df_grouped['com_pos_ctc_mean'] > 0]
df_grouped.head()
```

[48]:

	company_hash	job_position	com_pos_ctc_mean
24	0	Others	100000.0
65	0000	Others	300000.0
86	01 ojztqsj	Android Engineer	270000.0
99	01 ojztqsj	Frontend Engineer	830000.0
130	05mz exzytvrny uqxcvnt rxbxnta	Backend Engineer	1100000.0

[49]:
```
df = pd.merge(df, df_grouped, on=['company_hash', 'job_position'], how='left')
df['Class'] = df.apply(lambda x: category_mapping(x['ctc'],␣
    ↪x['com_pos_ctc_mean']), axis=1)
df.drop(columns=['com_pos_ctc_mean'], inplace=True)
```

[50]:
```
df_grouped = df.groupby(['company_hash'])['ctc'].mean().reset_index()
df_grouped.rename(columns={'ctc': 'com_ctc_mean'}, inplace=True)
```

```
df_grouped = df_grouped[df_grouped['com_ctc_mean'] > 0]
df_grouped.head()
```

[50]:

```
                         company_hash  com_ctc_mean
0                                   0      100000.0
1                                0000      300000.0
2                          01 ojztqsj      550000.0
3  05mz exzytvrny uqxcvnt rxbxnta     1100000.0
4                                   1      175000.0
```

[51]:

```
df = pd.merge(df, df_grouped, on=['company_hash'], how='left')
df['Tier'] = df.apply(lambda x: category_mapping(x['ctc'], x['com_ctc_mean']),␣
 ↪axis=1)
df.drop(columns=['com_ctc_mean'], inplace=True)
```

[52]:

```
df.sample(5)
```

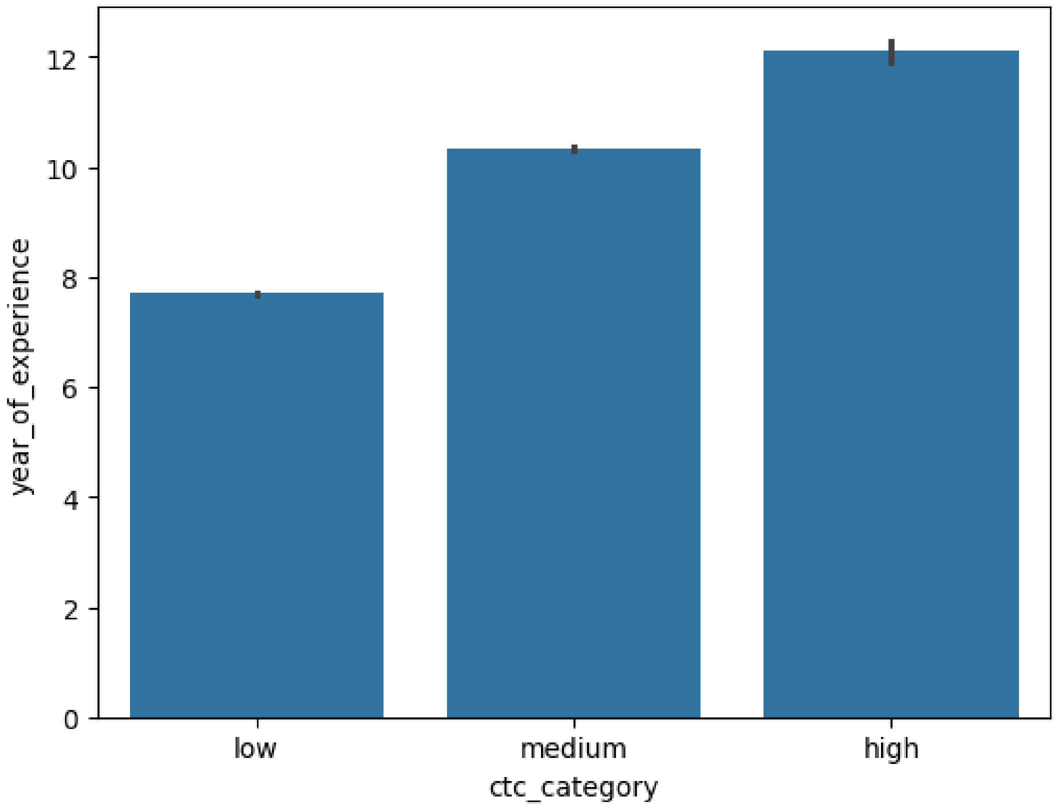
[52]:

```
                                   company_hash  \
87394                   ztdngqj uqxcvnt rxbxnta
74538  tdwtrrtd ntwyzgrgsxto uqxcvnt rxbxnta
27458                            ti ntwyzgrgsxw
42287                              ojzguojo xzw
35043                                       zvz

                                              email_hash  orgyear      ctc  \
87394  81f9e9d24d143e18a6b4e4ea64792038392ca1d5f32f5a…     2019   550000
74538  6ed3aa0728ec2819fd74bb9a56e94121e9affa4ed238ce…     2015   672000
27458  286e211d9ac97b8f644751c955614bdac0056f4bc3cea0…     2017  1500000
42287  3ea8a32a8fa5f84afbb931771154609428bb29065d00d3…     2020   380000
35043  33e1fc8bd06fc0f20c643df27a2667bcad34d2d6ba66fb…     2015  2200000

            job_position  ctc_updated_year  CTC_zscore ctc_category  \
87394  Frontend Engineer              2021   -0.149401          low
74538  Fullstack Engineer             2021   -0.139541          low
27458  Frontend Engineer              2021   -0.072625       medium
42287   Backend Engineer              2019   -0.163140          low
35043  Frontend Engineer              2021   -0.016053       medium

       year_of_experience  designation  Class  Tier
87394                   5            2      2     2
74538                   9            2      2     2
27458                   7            3      3     3
42287                   4            1      1     1
35043                   9            3      3     3
```

[53]:

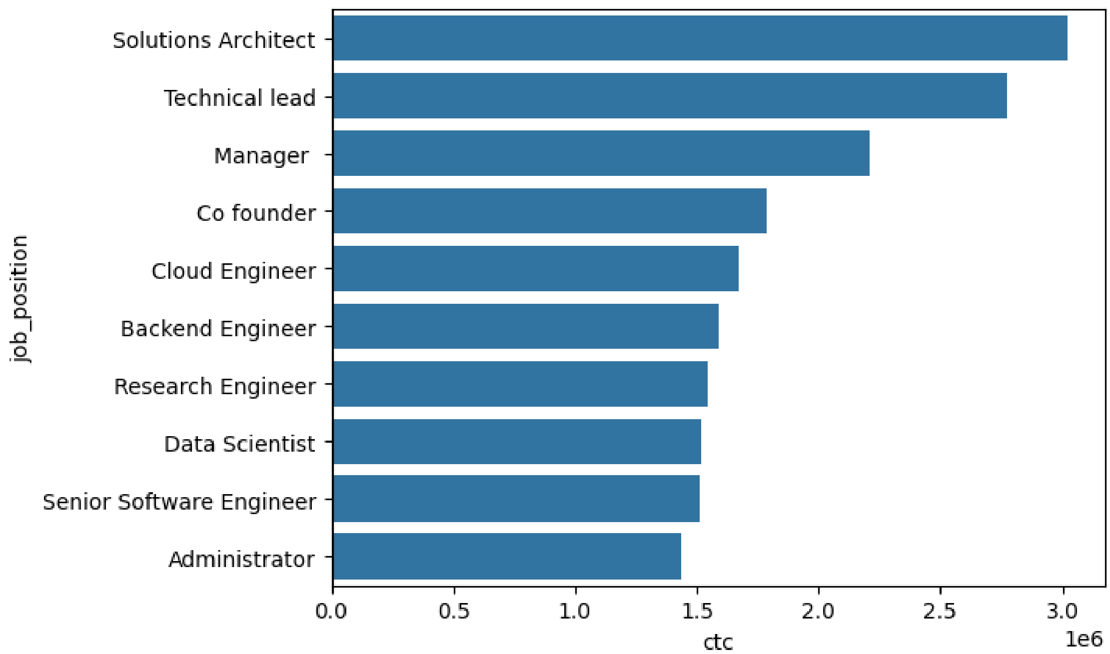
```
sns.barplot(x='ctc_category', y='year_of_experience', data=df)
```

[53]: <Axes: xlabel='ctc_category', ylabel='year_of_experience'>



[54]:
```python
# Top 10 Job positions with highest mean CTC
top_10_job_positions = df.groupby('job_position')['ctc'].mean().
 ↪sort_values(ascending=False).head(10).reset_index()
sns.barplot(x='ctc', y='job_position', data=top_10_job_positions)
```

[54]: <Axes: xlabel='ctc', ylabel='job_position'>



0.1.13 Insights:

Top 10 Job positions with highest mean CTC * Solutions Architect * Technical lead * Manager * Co founder * Cloud Engineer * Backend Engineer * Research Engineer * Data Scientist * Senior Software Engineer * Administrator

[55]:
```
df[df['Tier']==1].sort_values(by='ctc', ascending=False).head(10)
```

[55]:
```
                                          company_hash  \
70681                                         aggqavoy
62401   mqxwponttr tzntquqxoto uqxcvnt rxbxnta
44445                        bsb qtogqno xzntqzvnxgzvr
31001                          zvnxgzvr vhonqvrxv mvzp
42591                                  xzagqot unt rna
141931                                       wgzerhtzn
118650                                    uqgutqnjshqh
17465                                           wvqttb
103987                                          uvd vx
103730                                          xatvrg

                                                  email_hash  orgyear       ctc  \
70681   68f1fea4dbfb7ae2209664b93d5f57fb86912dbe516b37...     2018  13500000
62401   5cb0417e963b2bb218dc28cbe0c9003c39c4f2db94bb53...     2016  10100000
44445   420388fd953332be671e1b0761f9af06d323382d075ecf...     2017   7000000
31001   2ddbc233754a1bf09fa7e92d61a5fb8fd46f3fe7908318...     2000   7000000
42591   3f0f3d507158974626454f18f97876cd1f3ffe816a9757...     2008   6500000
```

```
141931  d2baa62a1b611ad438f0475c2e3d88f2c6f9f3df6e56bf…     2015    6000000
118650  b01a6b018bb4ce949e745045fae952bd9c89566480e066…     2015    5820000
17465   19839c1100097c37582a06a4dbb8ec28194883f0c6b70f…     2017    5600000
103987  9a6853551e4a153f057d06ae57bfba4a7027cdf43cf273…     2012    5600000
103730  9a060e939ffc6ec3f5a9351ffb303ad6d064761ee66c94…     2000    5440000

             job_position  ctc_updated_year  CTC_zscore ctc_category  \
70681     Backend Engineer              2020    0.897179         high
62401          QA Engineer              2019    0.622401         high
44445    Fullstack Engineer             2018    0.371868         high
31001       Technical lead              2020    0.371868         high
42591     Backend Engineer              2020    0.331460         high
141931    Backend Engineer              2020    0.291051         high
118650  Fullstack Engineer              2020    0.276504         high
17465     Backend Engineer              2020    0.258725         high
103987    Backend Engineer              2019    0.258725         high
103730      Technical lead              2020    0.245794         high

        year_of_experience  designation  Class  Tier
70681                    6            2      2     1
62401                    8            1      1     1
44445                    7            2      2     1
31001                   24            2      2     1
42591                   16            1      1     1
141931                   9            2      2     1
118650                   9            2      1     1
17465                    7            1      1     1
103987                  12            2      2     1
103730                  24            2      2     1
```

[56]:

```python
# Top 10 employees of data science in each company earning more than their peers - Class 1
top_10_employees = df[(df['job_position']=='Data Scientist') & (df['Class']==1)]
top_10_employees = top_10_employees.groupby('company_hash').apply(lambda x: x.nlargest(10, 'ctc')).reset_index(drop=True)
top_10_employees.sort_values(by='company_hash')
```

[56]:

```
                     company_hash  \
0                             1bs
1                           247vx
2                           247vx
3                  3p ntwyzgrgsxto
4                  3p ntwyzgrgsxto
…                               …
1491             zxxn ntwyzgrgsxto
1493  zxxn ntwyzgrgsxto rxbxnta
1495                     zxztrtvuo
```

```
1494                   zxztrtvuo
1496                   zxztrtvuo

                                               email_hash  orgyear      ctc  \
0     eb213c0552effd7fb139395c7838edb8d59773a1cb57a0…     1994   800000
1     c35054c043f6a02da3e6f142fbcb095f8145eb521137ff…     2014  2150000
2     5f4b52a1c2539fe2e4b29a8470bc57dbace331b819a0af…     2002  1440000
3     583a9600d8e74d5f3f6627da6b4ff3c466bf5cfee5ae9f…     2018  1200000
4     f3b1e96456ad8a7d9cceb3901d763252ea2f56eb2ee7f4…     2013  1200000
…                                                     …      …      …
1491  1aa2717970a46b5d12b90932799227774dd418c842fa18…     2012  2200000
1493  5ab93fd511bceaa6da5f855d160de306a04df9951f5978…     2012   800000
1495  3027ca561b65f99da2f65bf3d85c6bb5d5687c67e69e89…     2018  1370000
1494  10d566c5fca40ffe1d133b79594d071880711ef480da9f…     2017  1400000
1496  f678c67bee8cad9370f6aaf4f4cc22ffd417fd753663c6…     2019  1250000

        job_position  ctc_updated_year  CTC_zscore ctc_category  \
0     Data Scientist              2019   -0.129197          low
1     Data Scientist              2018   -0.020094       medium
2     Data Scientist              2019   -0.077474       medium
3     Data Scientist              2019   -0.096870       medium
4     Data Scientist              2019   -0.096870       medium
…                  …                 …           …          …
1491  Data Scientist              2019   -0.016053       medium
1493  Data Scientist              2021   -0.129197          low
1495  Data Scientist              2019   -0.083131       medium
1494  Data Scientist              2019   -0.080707       medium
1496  Data Scientist              2021   -0.092829       medium

      year_of_experience  designation  Class  Tier
0                     30            2      1     1
1                     10            2      1     3
2                     22            2      1     1
3                      6            1      1     3
4                     11            2      1     3
…                    …            …      …     …
1491                  12            2      1     3
1493                  12            2      1     1
1495                   6            2      1     3
1494                   7            2      1     3
1496                   5            2      1     3

[1497 rows x 12 columns]
```

[57]:

```
# Bottom 10 employees of data science in each company earning less than their␣
↪peers - Class 3
bottom_10_employees = df[(df['job_position']=='Data Scientist') &␣
↪(df['Class']==3)]

# Initialize an empty DataFrame to store the results
result_df = pd.DataFrame()

# Process each company separately to avoid memory issues
for company in bottom_10_employees['company_hash'].unique():
        company_df = bottom_10_employees[bottom_10_employees['company_hash'] ==␣
↪company]
        bottom_10 = company_df.groupby('email_hash')['ctc'].mean().
↪nsmallest(10).reset_index()
        bottom_10['company_hash'] = company
        result_df = pd.concat([result_df, bottom_10], ignore_index=True)

result_df = result_df[result_df['ctc'].notnull()]
result_df
```

[57]:

```
                                             email_hash        ctc  \
0     63ea6962348c633656fb3e335c41356c524ed6559176a7…  1610000.0
1     001f9dbae7fccf1ef52c2187359d65c17e2897b8d211d2…  2000000.0
2     a59dd7dfe78bdb896a0aa81b0fa938cd4459a99f14fa03…  2000000.0
10    0038257f6ef2580dfbc8566cc80d519819c97c2dcbddde…  4200000.0
20    70e3f9611e8afdd1184c174d0948575c6fc3dba0b1aa21…  1050000.0
…                                                   …        …
6340  fb5db1053d6c3d953112bd6316b9d0ebe2954dc4df3c78…  3500000.0
6350  fc0a94acd05743069d35b4dab69efd913fce055f374c8f…  2500000.0
6360  fc48d558c7d9abc5b8ca58c524173255b26782c15c28b9…  2000000.0
6370  fd97ee5909e0db824aaf15b5f293987a1c21272ecc491e…  5600000.0
6380  ffb56190424c95d9b895e9e3712cdc94c4382011357d18…  1100000.0

                      company_hash
0                      xzwtag xzw
1                      xzwtag xzw
2                      xzwtag xzw
10    rn ntwyzgrgsj otqcxwto rxbxnta
20                              tj
…                              …
6340                    ovrtoegqwt
6350                      ktzaqxct
6360                         ihxpq
6370                   jghurho xzw
6380          vrsgzgd ntwyzgrgsxto

[1261 rows x 3 columns]
```

[58]:
```python
# Bottom 10 employees (earning less than most of the employees in the company)␣
↪- Tier 3
bottom_10_employees = df[df['Tier']==3]

# Initialize an empty DataFrame to store the results
result_df = pd.DataFrame()

# Process each company separately to avoid memory issues
for company in bottom_10_employees['company_hash'].unique():
        company_df = bottom_10_employees[bottom_10_employees['company_hash'] ==␣
↪company]
        bottom_10 = company_df.groupby('email_hash')['ctc'].mean().
↪nsmallest(10).reset_index()
        bottom_10['company_hash'] = company
        result_df = pd.concat([result_df, bottom_10], ignore_index=True)

result_df = result_df[result_df['ctc'].notnull()]
result_df.sort_values(by='ctc', ascending=True).head(10)
```

[58]:
```
                                              email_hash       ctc  \
82840  df98a0f24fb45172e2768b53515d980f2ff5b40265d8b8…  105000.0
65050  979c186a6cd175b2dfbfea8e50d3bb823bb4fb9d2fd5e5…  110000.0
33330  369ccf8ed6bf03c964d2822f76119ba94776b0c9ad7485…  115000.0
15510  110f894dcc2dea3cbdcbd8f6e5857407184c9062920a30…  115000.0
75360  c0e59bea31301b6f14bcb8bf8ff87485ffdeac52a5d5f3…  120000.0
900    00972ad5cc152a8763c501413f1213b4c0c9cb78ed735b…  120000.0
83090  e12764654b74f407aa9ddf4819030c1d9cc88e9c3f359f…  120000.0
33331  9c8846c710a0266e74dd7d29e0d3d61f533a0da533e1ac…  120000.0
75600  c1b86ff295f82151e1649a338eaae941766375a74fca96…  120000.0
78290  cc51b85d8792b7755d107504dfa1ffa6123ade28c65bd7…  120000.0

                    company_hash
82840            eqjo trtwnqgzxwo
65050            ctqxkgz fxqtrtoo
33330              oyhnntqerj xzw
15510                wxqwrto rxet
75360        uvoohq vtqgouvwt xzw
900    ovzaxv zvnxgzvr rvmgqvngqxto
83090    uwo srgmvr uqxcvnt rxbxnta
33331              oyhnntqerj xzw
75600                 xytvqnqvaxg
78290                  tuxw svbto
```

[59]:
```python
# Top 10 employees in each company - X department - having 5/6/7 years of␣
↪experience earning more than their peers - Tier X

# Initialize an empty DataFrame to store the results
```

```
result_df = pd.DataFrame()

# Filter the relevant data
filtered_df = df[
    (df['job_position']=='Data Scientist') & (df['year_of_experience'].
 ↪isin([5,6,7])) & (df['Tier']==1)
]

# Process each company separately to avoid memory issues
for company in filtered_df['company_hash'].unique():
    company_df = filtered_df[filtered_df['company_hash'] == company]
    top_10 = company_df.groupby('email_hash')['ctc'].mean().nlargest(10).
 ↪reset_index()
    top_10['company_hash'] = company
    result_df = pd.concat([result_df, top_10], ignore_index=True)

result_df = result_df[result_df['ctc'].notnull()]
result_df.sort_values(by='ctc', ascending=True).head(10)
```

[59]:

```
                                            email_hash       ctc  \
2980  6d50e60d562940b28dc7cd0296ea52164d272d7f5696f8…  100000.0
3321  7ff23b1d29cb8a2546289c567848c6f2ebbcd335af9f5c…  100000.0
1317  2cadff2fe2f39e9e4e684b95c0db58c9344fceb815c6c1…  100000.0
1820  3f1cd17776c41c8395107a340b76af3a56fe307987e271…  103000.0
400   0a65cc19cc9402203e3282318b9a4a127eb0b6691120cc…  110000.0
1430  30b64da7d7e52c133a0e1df85c912794081242e0a792fa…  110000.0
5360  fa24a6efde82936d129f71557a1723fcefcf0db9eee432…  110000.0
3320  a9560804241d30092205375646cd9c0df1c4a49204e678…  112000.0
4800  db2c70fea469a7f1456457812fe94a01c337eb6ce75bd5…  115000.0
1670  38b81aab25e388fb3282b99e6872365cf15b80a08bd787…  120000.0

                 company_hash
2980                   ktgnvu
3321  ytrrgoxcx ogenfvqt rvmo
1317                     zgzt
1820               hztburgjta
400                  ihvznuyx
1430              vhaxmrt xzw
5360               xzatrrxtzn
3320  ytrrgoxcx ogenfvqt rvmo
4800                    mvjtq
1670                  xzwgqnv
```

[60]:

```
# Initialize an empty DataFrame to store the results
result_df = pd.DataFrame()

# Filter the relevant data
```

```
filtered_df = df[
    (df['job_position']=='Data Scientist') & (df['year_of_experience'].
 ↪isin([5,6,7])) & (df['Tier']==3)
]

# Process each company separately to avoid memory issues
for company in filtered_df['company_hash'].unique():
    company_df = filtered_df[filtered_df['company_hash'] == company]
    top_10 = company_df.groupby('email_hash')['ctc'].mean().nlargest(10).
 ↪reset_index()
    top_10['company_hash'] = company
    result_df = pd.concat([result_df, top_10], ignore_index=True)

result_df = result_df[result_df['ctc'].notnull()]
result_df
```

[60]:

```
                                              email_hash        ctc  \
0     009ded427ebcb5c2fb1970017a683693a7abef0fa96f5e…  3900000.0
1     ea7eb74cbbabcb83aa2f66891c6deb847a603c2a40535f…  2800000.0
2     10dc906ae6fedaac7e16b0dd71356be39e2136911990a9…  2380000.0
3     3fdfb9b1014a0d22fd5fa9d5c7a686a31a16a933eda144…  2300000.0
4     aedf693542ac76087cb458cfc66154382568d02f0d6acb…  2200000.0
…                                                   …          …
3890  fba008a8a93022172e3adfa293d9ce0f4d70267afa373d…   500000.0
3900  fbac8d7a3768fdb72d929a641d9ec1c307d35487bd0aad…  1500000.0
3910  fbb5702400e9542c9ddcbb561189dafec71131f8458856…  1050000.0
3920  fc48d558c7d9abc5b8ca58c524173255b26782c15c28b9…  2000000.0
3930  ffc974693a2bfd0326c707d8460d6783861a9497e538e2…  1700000.0

                          company_hash
0                     eqvwnvr vzvrjnxwo
1                     eqvwnvr vzvrjnxwo
2                     eqvwnvr vzvrjnxwo
3                     eqvwnvr vzvrjnxwo
4                     eqvwnvr vzvrjnxwo
…                                     …
3890                        vuurxta vx
3900  bvyxzaqv wgbcxcv ntwyzgrgsxto rna
3910                          bvqgugon
3920                             ihxpq
3930                   atrgxnnt xzaxv

[566 rows x 3 columns]
```

[61]:

```
# Initialize an empty DataFrame to store the results
result_df = pd.DataFrame()
```

```
# Filter the relevant data
filtered_df = df[
    (df['job_position']=='Data Scientist') & (df['year_of_experience'].
 ↪isin([5,6,7])) & (df['Tier']==2)
]

# Process each company separately to avoid memory issues
for company in filtered_df['company_hash'].unique():
    company_df = filtered_df[filtered_df['company_hash'] == company]
    top_10 = company_df.groupby('email_hash')['ctc'].mean().nlargest(10).
 ↪reset_index()
    top_10['company_hash'] = company
    result_df = pd.concat([result_df, top_10], ignore_index=True)

result_df = result_df[result_df['ctc'].notnull()]
result_df
```

[61]:

```
                                             email_hash        ctc  \
0     013a80ea3b5799eb374ec5dd94d3c84579ba0b44293258…   400000.0
10    01e291d69add1b6340ff19d8f4cb396ffb2a7a58c5b82b…   900000.0
20    01e29469795e1395f058e33226e15b1a32d335e92ac5bf…  1800000.0
30    0272dcd1909235f14140851a86a1bfb22098d4c71bd0f7…  1500000.0
40    02d70fdfcde30937fa1953b7e17b4f20b6bb57de684ec5…   700000.0
…                                                   …          …
3600  fa57641000b5bb092b93eb2d5b615275b550045aece5b9…   750000.0
3610  fb62ad17e33d701e2369045284190ff75e7e795eba1dc5…   620000.0
3620  fc0472f7b02f8569d8a368972f2f6f99e509a961e9078e…  1980000.0
3630  ff7bace89454b0965e40533aa5b7f904dba9c496b1a1f2…  1019999.0
3640  ffa4d3df725be3628607627e575feac064a1e98506dc19…   900000.0

                                   company_hash
0                           mxsnvuu vzvrjnxwo
10            avnv owxtzwt nqvxztt vn tafxogq
20                 tuxex ntwyzgrgsxto ucn rna
30                                       vqjv
40                                      vqjzs
…                                           …
3600          bvszv xzegntwy  v ihtoo wgbuvzj
3610         vae avnv owxtzwt uqxcvnt rxbxnta
3620       l u bgqsvz otqcxwto xzaxv ucn rna
3630  xuqtaxwnwgszxnxct ntwyzgrgsxto ucn rna
3640                                    qtmxn

[366 rows x 3 columns]
```

[62]:

```python
# Top 10 companies (based on their CTC)
top_10_companies = df.groupby('company_hash')['ctc'].mean().
    sort_values(ascending=False).head(10).reset_index()
top_10_companies
```

[62]:

	company_hash	ctc
0	mqvojo	15000000.0
1	qtuogr	15000000.0
2	ihxu	15000000.0
3	cyhm	15000000.0
4	qtvrotre	15000000.0
5	hzxctqoxnj ge ntdvo vn avrrvo	15000000.0
6	vnox xzw	14400000.0
7	ogenfvqt tzsxzttqxzs	14400000.0
8	gqvmvot ogrhnxgzo	14400000.0
9	xnc	14000000.0

[63]:

```python
# Top 2 positions in every company (based on their CTC)
top_2_positions = df.groupby(['company_hash', 'job_position'])['ctc'].mean().
    groupby('company_hash', group_keys=False).nlargest(2).reset_index()
top_2_positions.groupby('company_hash')['job_position'].apply(list).
    reset_index()
```

[63]:

	company_hash	job_position
0	0	[Others, Accounting]
1	0000	[Others, Accounting]
2	01 ojztqsj	[Frontend Engineer, Android Engineer]
3	05mz exzytvrny uqxcvnt rxbxnta	[Backend Engineer, Accounting]
4	1	[Others, Frontend Engineer]
…	…	…
37294	zyvzwt wgzohrnxzs tzsxzttqo	[Frontend Engineer, Accounting]
37295	zz	[Others, Frontend Engineer]
37296	zzb ztdnstz vacxogqj ucn rna	[Fullstack Engineer, Accounting]
37297	zzgato	[Frontend Engineer, Accounting]
37298	zzzbzb	[Others, Accounting]

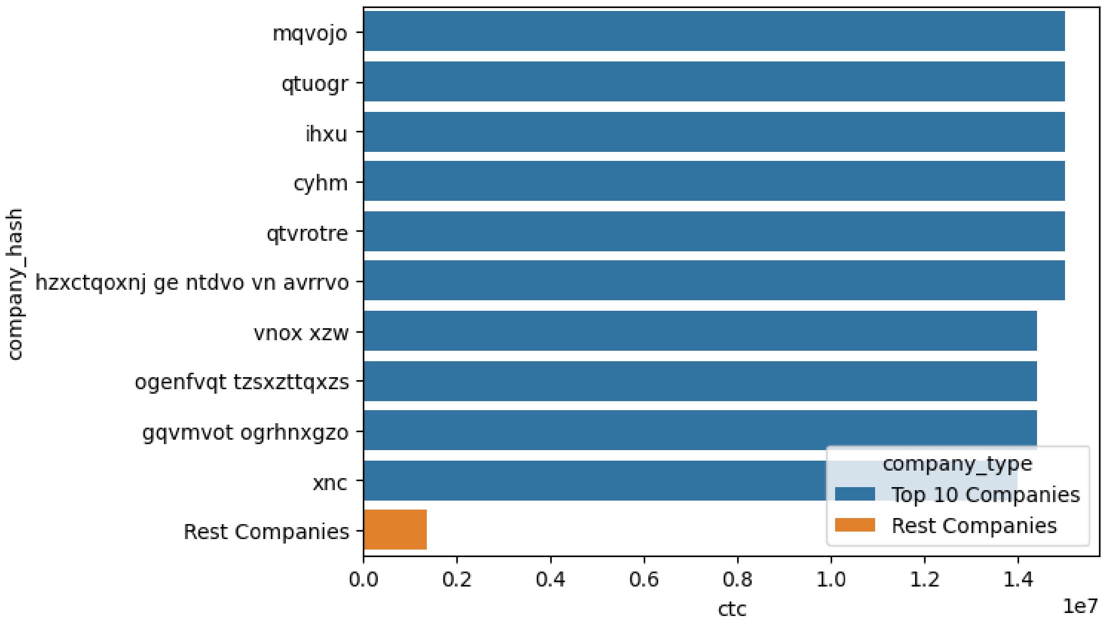
[37299 rows x 2 columns]

[64]:

```python
# compare the mean CTC of the top 10 companies with the mean CTC of the rest of
    the companies
top_10_companies = df.groupby('company_hash')['ctc'].mean().
    sort_values(ascending=False).head(10).reset_index()
top_10_companies['company_type'] = 'Top 10 Companies'

rest_companies = df[~df['company_hash'].isin(top_10_companies['company_hash'])]
rest_companies_mean_ctc = rest_companies['ctc'].mean()
```

```python
rest_companies = pd.DataFrame({'company_hash': ['Rest Companies'], 'ctc':␣
↪[rest_companies_mean_ctc], 'company_type': ['Rest Companies']})

companies_df = pd.concat([top_10_companies, rest_companies], ignore_index=True)
sns.barplot(x='ctc', y='company_hash', data=companies_df, hue='company_type')
```

[64]: <Axes: xlabel='ctc', ylabel='company_hash'>



[65]:
```python
df.sample(5)
```

[65]:
```
                                   company_hash  \
150583                                    kxusg
112825                                   rtznqv
62856                                      exwg
25746   uqvbvnx ntwyzgrgsxto uqxcvnt rxbxnta
133659                                   whqgej

                                               email_hash  orgyear      ctc  \
150583  dfa84e64857e65fb415417939f6393f520462cae373a4f…     2014  1700000
112825  a787c58d015cfb352a1d4244d52da45738169326b999f2…     2018   660000
62856   5d6d61c7f0a99ff6fc574b8ed0263eb3132d060cfc575e…     2010  2000000
25746   25dfdf2bbb2b5f45fb5f82ed2d7620f7b25806431df906…     2014   600000
133659  c682f7df08f21debe505a5eaca754e05b79b5a6a17e060…     2010   700000

           job_position  ctc_updated_year  CTC_zscore ctc_category  \
150583          Manager              2019   -0.056461       medium
```

```
112825   Backend Engineer              2021   -0.140511         low
62856    Backend Engineer              2019   -0.032216      medium
25746         QA Engineer              2019   -0.145360         low
133659  Frontend Engineer              2019   -0.137278         low

        year_of_experience  designation  Class  Tier
150583                  10            2      1     3
112825                   6            2      2     3
62856                   14            3      3     3
25746                   10            2      2     1
133659                  14            2      1     1
```

0.1.14 Data preparation for Modelling

[74]:
```
scaled_df = df.copy()
```

[75]:
```
# Standardize Numerical Data
numerical_cols = ['ctc', 'year_of_experience', 'orgyear','ctc_updated_year']
scaler = StandardScaler()
scaled_df[numerical_cols] = scaler.fit_transform(scaled_df[numerical_cols])

# Frequency Encoding for company_hash
company_hash_frequency = scaled_df['company_hash'].value_counts().to_dict()
scaled_df['company_hash_encoded'] = scaled_df['company_hash'].
↪map(company_hash_frequency)

# Encode Other Categorical Data
categorical_cols = ['job_position', 'ctc_category', 'Class', 'Tier']
encoder = OneHotEncoder(sparse_output=False)
encoded_categorical_data = encoder.fit_transform(scaled_df[categorical_cols])
encoded_categorical_df = pd.DataFrame(encoded_categorical_data, columns=encoder.
↪get_feature_names_out(categorical_cols))

# Concatenate the encoded categorical data with the original DataFrame
scaled_df = pd.concat([scaled_df, encoded_categorical_df], axis=1)

# Drop the original categorical columns
scaled_df.drop(columns=categorical_cols, inplace=True)
scaled_df.drop(columns=['company_hash','CTC_zscore','email_hash'], inplace=True)
```

[76]:
```
scaled_df.sample(5)
```

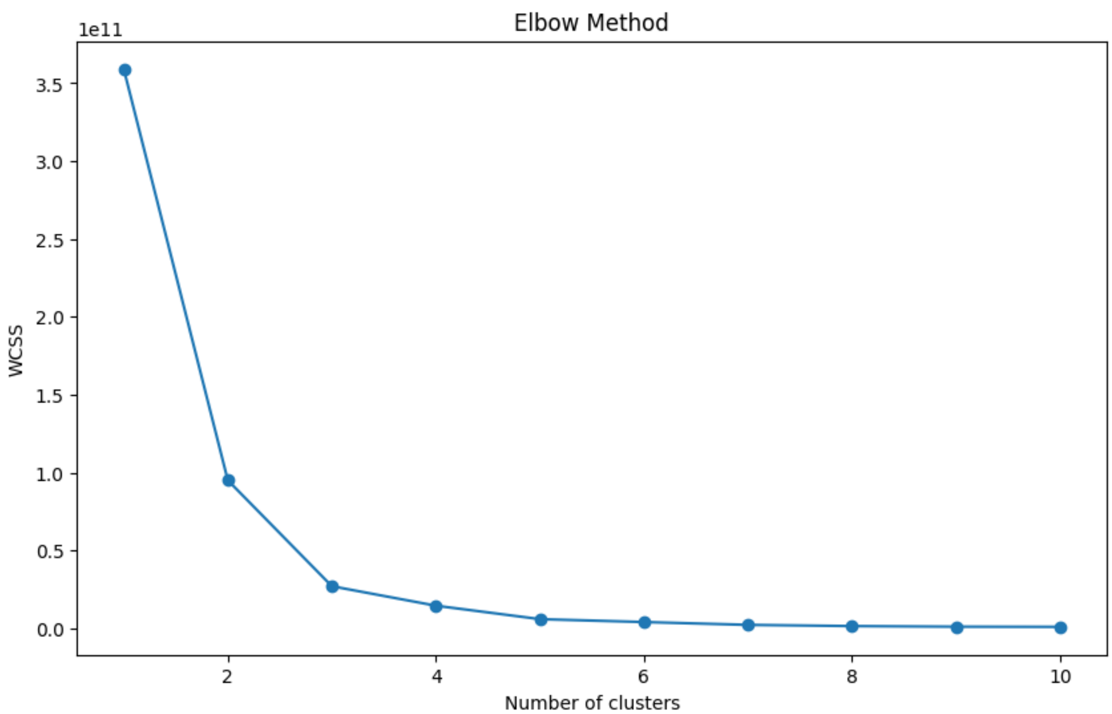
[76]:
```
         orgyear       ctc  ctc_updated_year  year_of_experience  designation  \
128407  0.465676  0.581414          1.115702           -0.465676            3
132177  0.230061 -0.665794          1.115702           -0.230061            2
```

```
142402 -1.890475  0.252396        -0.388753           1.890475            2
144312 -0.712400 -0.589278         1.115702           0.712400            1
108296 -0.948015 -0.359730        -1.893209           0.948015            3

        company_hash_encoded  job_position_Accounting  \
128407                  1459                      0.0
132177                     2                      0.0
142402                    52                      0.0
144312                    41                      0.0
108296                   288                      0.0

        job_position_Administrator  job_position_Advisory  \
128407                         0.0                    0.0
132177                         0.0                    0.0
142402                         0.0                    0.0
144312                         0.0                    0.0
108296                         0.0                    0.0

        job_position_Analyst  …  job_position_iOS Engineer  \
128407                   0.0  …                        0.0
132177                   0.0  …                        0.0
142402                   0.0  …                        0.0
144312                   0.0  …                        0.0
108296                   0.0  …                        0.0

        ctc_category_high  ctc_category_low  ctc_category_medium  Class_1  \
128407                0.0               0.0                  1.0      0.0
132177                0.0               1.0                  0.0      0.0
142402                0.0               0.0                  1.0      0.0
144312                0.0               1.0                  0.0      1.0
108296                0.0               1.0                  0.0      0.0

        Class_2  Class_3  Tier_1  Tier_2  Tier_3
128407      0.0      1.0     0.0     0.0     1.0
132177      1.0      0.0     0.0     0.0     1.0
142402      0.0      1.0     0.0     0.0     1.0
144312      0.0      0.0     1.0     0.0     0.0
108296      0.0      1.0     1.0     0.0     0.0

[5 rows x 56 columns]
```

Model Building

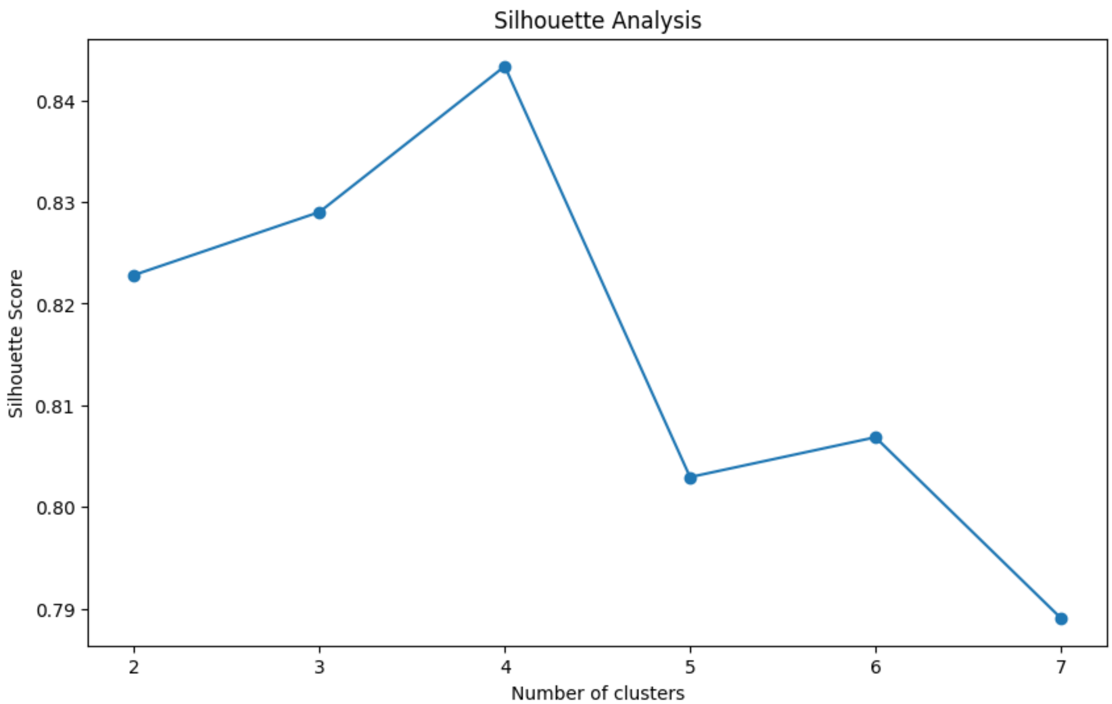
KMeans Clustering

[77]:
```
X = scaled_df.values

# Elbow Method
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), wcss, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



[78]:
```
# Silhouette Analysis
silhouette_scores = []
for i in range(2, 8):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(X)
    score = silhouette_score(X, kmeans.labels_)
    silhouette_scores.append(score)
```

```
plt.figure(figsize=(10, 6))
plt.plot(range(2, 8), silhouette_scores, marker='o')
plt.title('Silhouette Analysis')
plt.xlabel('Number of clusters')
plt.ylabel('Silhouette Score')
plt.show()
```



0.1.15 Elbow Method and Silhouette Analysis

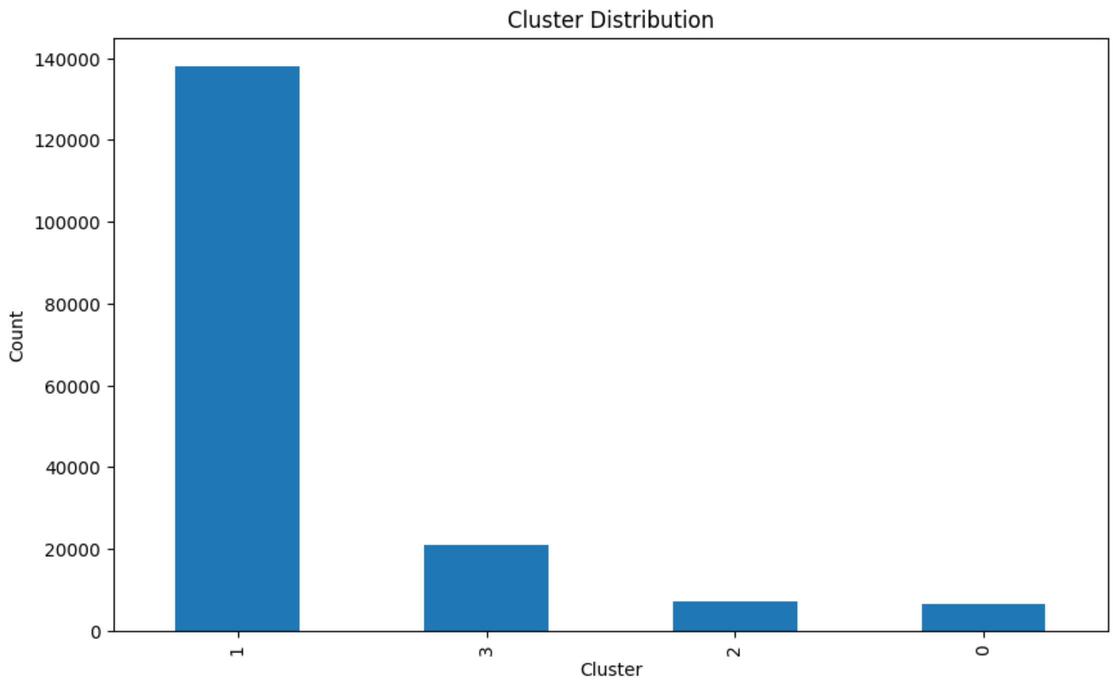
- **Elbow Method:** Suggests a potential optimal number of clusters at 3, as there is a noticeable drop in WCSS.
- **Silhouette Analysis:** Indicates that 4 clusters have the highest silhouette score, suggesting the best-defined clusters.

Considering both methods, the optimal number of clusters appears to be 4 clusters. While the Elbow Method suggests 3 clusters, the Silhouette Analysis strongly indicates that 4 clusters provide the best-defined clusters. Therefore, 4 clusters are likely the most appropriate choice for this dataset.

[118]:
```
kmeans = KMeans(n_clusters = 4, random_state = 3)
kmeans.fit(X)

scaled_df['kM_cluster'] = kmeans.labels_
scaled_df['kM_cluster'].value_counts()
```

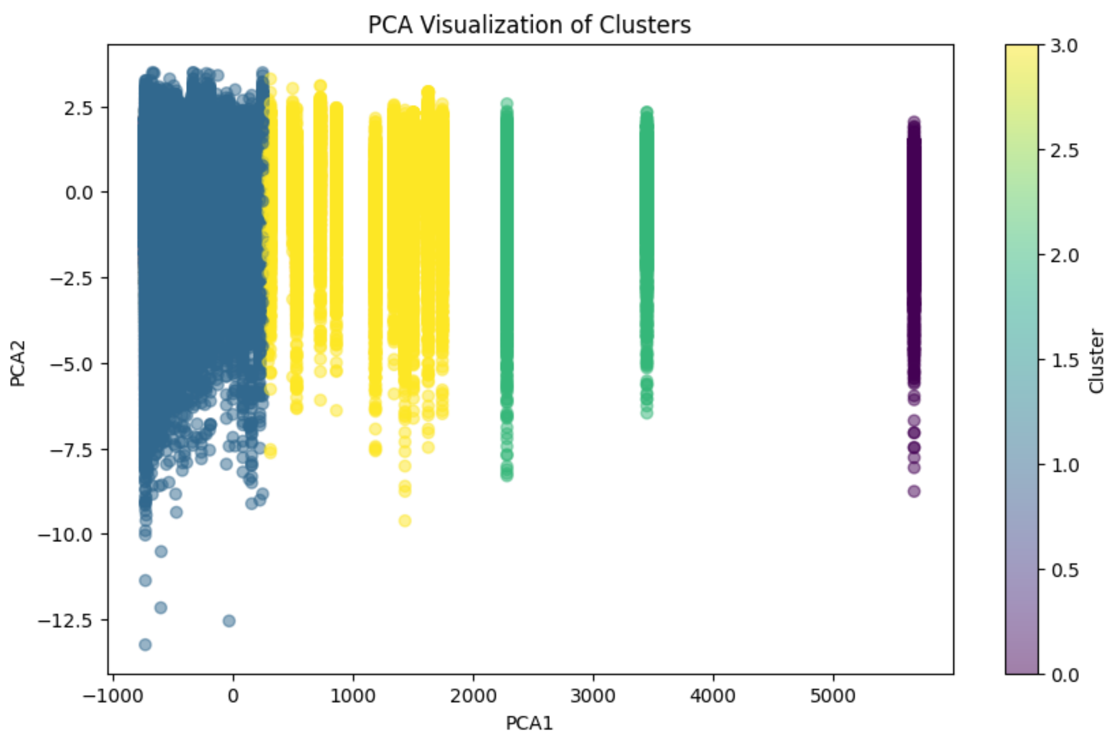
[118]: kM_cluster
```
1    138028
3     20824
2      7189
0      6402
Name: count, dtype: int64
```

[119]:
```
# Distribution of each cluster
plt.figure(figsize=(10, 6))
scaled_df['kM_cluster'].value_counts().plot(kind='bar')
plt.title('Cluster Distribution')
plt.xlabel('Cluster')
plt.ylabel('Count')
plt.show()
```



[120]:
```
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

# Perform PCA to reduce dimensionality to 2D for visualization
pca = PCA(n_components=2)
pca_result = pca.fit_transform(scaled_df.drop(columns=['kM_cluster']))

# Create a DataFrame with PCA results and cluster labels
pca_df = pd.DataFrame(pca_result, columns=['PCA1', 'PCA2'])
pca_df['Cluster'] = scaled_df['kM_cluster']
```

```
# Plot the PCA results
plt.figure(figsize=(10, 6))
plt.scatter(pca_df['PCA1'], pca_df['PCA2'], c=pca_df['Cluster'],␣
↪cmap='viridis', alpha=0.5)
plt.title('PCA Visualization of Clusters')
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.colorbar(label='Cluster')
plt.show()
```



[127]:
```
df['kM_cluster'] = kmeans.labels_
print('-'*5, 'Cluster Analysis', '-'*5)
print(df.groupby('kM_cluster')['year_of_experience'].mean())
print('-'*10, 'Cluster Mean', '-'*10)
print(df.groupby('kM_cluster')['ctc'].mean())
```

```
----- Cluster Analysis -----
kM_cluster
0    6.785848
1    9.271662
2    7.678398
3    8.141087
Name: year_of_experience, dtype: float64
```

```
---------- Cluster Mean ----------
kM_cluster
0    6.138651e+05
1    1.418803e+06
2    1.366223e+06
3    1.281431e+06
Name: ctc, dtype: float64
```

[128]:

```
print('-'*10, 'Cluster Median', '-'*10)
print(df.groupby('kM_cluster')['ctc'].median())
print('-'*10, 'Cluster Minimum', '-'*10)
print(df.groupby('kM_cluster')['ctc'].min())
print('-'*10, 'Cluster Maximum', '-'*10)
print(df.groupby('kM_cluster')['ctc'].max())
```

```
---------- Cluster Median ----------
kM_cluster
0     450000.0
1    1040000.0
2     700000.0
3     819999.0
Name: ctc, dtype: float64
---------- Cluster Minimum ----------
kM_cluster
0    100000
1    100000
2    100000
3    100000
Name: ctc, dtype: int32
---------- Cluster Maximum ----------
kM_cluster
0    11200000
1    15000000
2    14700000
3    15000000
Name: ctc, dtype: int32
```

0.1.16 Interpretation

Cluster 0: - **Median CTC:** This cluster has the lowest median CTC at 450,000, indicating that the majority of individuals in this cluster have relatively lower salaries compared to other clusters. - **Range:** The CTC ranges from 100,000 to 11,200,000, showing a significant spread. This suggests that while the median is low, there are some high earners within this cluster.

Cluster 1: - **Median CTC:** This cluster has the highest median CTC at 1,040,000, indicating that the majority of individuals in this cluster have higher salaries compared to other clusters. - **Range:** The CTC ranges from 100,000 to 15,000,000, indicating a wide spread. This cluster includes both high earners and individuals with lower salaries.

Cluster 2: - **Median CTC:** This cluster has a median CTC of 700,000, which is moderate compared to other clusters. - **Range:** The CTC ranges from 100,000 to 14,700,000, showing a wide spread. This cluster includes a mix of individuals with varying salary levels.

Cluster 3: - **Median CTC:** This cluster has a median CTC of 819,999, which is slightly higher than Cluster 2 but lower than Cluster 1. - **Range:** The CTC ranges from 100,000 to 15,000,000, indicating a wide spread. This cluster also includes a mix of individuals with varying salary levels.

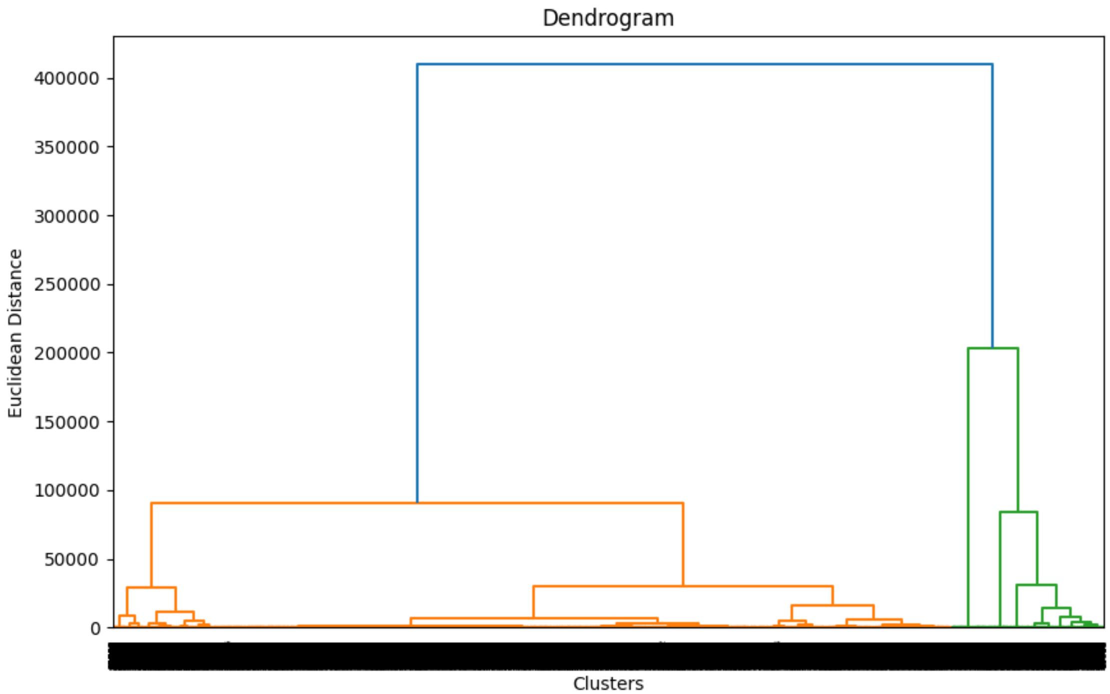
[132]:
```
scaled_df.shape
```

[132]:
```
(172443, 57)
```

0.1.17 Hierarchical Clustering

[135]:
```
sampled_df = scaled_df.sample(55000)
sampled_df = sampled_df.drop(columns=['kM_cluster'])
```

[136]:
```
# Hierarchical Clustering - Dendrogram

cluster_0 = sampled_df.values
Z = linkage(cluster_0, method='ward')

plt.figure(figsize=(10, 6))
dendrogram(Z)
plt.title('Dendrogram')
plt.xlabel('Clusters')
plt.ylabel('Euclidean Distance')
plt.show()
```



[144]:
```python
# Cut the dendrogram to form 3 clusters
num_clusters = 3
clusters = fcluster(Z, num_clusters, criterion='maxclust')

# Add the cluster labels to the subset DataFrame
sampled_df['Cluster'] = clusters
```

0.1.18 Actionable Insights and Recommendations

- **Targeted Interventions:**
  - For learners in the low category, consider providing additional training and resources to help them move to higher CTC categories.
  - For learners in the medium category, focus on career development and opportunities for promotions to help them achieve higher CTC values.
  - For learners in the high category, provide support for continued professional growth and retention strategies to maintain their high performance.
- **Role Transition:**
  - For learners experiencing the plateau effect, consider transitioning to higher-level roles (e.g., from software engineer to technical lead) to achieve further salary growth.
- **Negotiation Skills:**
  - Equip learners with negotiation skills to help them secure better salaries, especially when they have significant experience or specialized skills.
