```python
import pandas as pd
```

```python
data=pd.read_csv(r'https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/002/856/original/scaler_clustering.csv')
```

```python
data
```

	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
...	...	...	...	...	...	...	...
205838	206918	vuurt xzw	70027b728c8ee901fe979533ed94ffda97be08fc23f33b...	2008.0	220000	NaN	2019.0
205839	206919	husqvawgb	7f7292ffad724ebbe9ca860f515245368d714c84705b42...	2017.0	500000	NaN	2020.0
205840	206920	vwwgrxnt	cb25cc7304e9a24facda7f5567c7922ffc48e3d5d6018c...	2021.0	700000	NaN	2021.0
205841	206921	zgn vuurxwvmrt	fb46a1a2752f5f652ce634f6178d0578ef6995ee59f6c8...	2019.0	5100000	NaN	2019.0
205842	206922	bgqsvz onvzrtj	0bcfc1d05f2e8dc4147743a1313aa70a119b41b30d4a1f...	2014.0	1240000	NaN	2016.0

205843 rows × 7 columns

```python
data.isna().sum()
```

```
Unnamed: 0              0
company_hash           44
email_hash              0
orgyear                86
ctc                     0
job_position        52564
ctc_updated_year        0
dtype: int64
```

```python
def pk(ctc):
  if ctc>1000000 and ctc<=2000000 :
    return 'A'
  elif ctc>2000000 and ctc<=3000000:
    return 'B'
  elif ctc>3000000 and ctc<=4000000:
    return 'c'
  elif ctc>4000000 and ctc<=6000000:
    return 'C'
  elif ctc>6000000 and ctc<=8000000:
    return 'D'
  elif ctc>8000000 and ctc<=10000000:
    return 'E'
  else:
    return 'F'
from sklearn.impute import SimpleImputer
```

Start coding or generate with AI.

```python
s1=s.fit(data[['orgyear']])
data['orgyear']=s1.transform(data[['orgyear']])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-11-42322984091d> in <cell line: 1>()
----> 1 s1=s.fit(data[['orgyear']])
      2 data['orgyear']=s1.transform(data[['orgyear']])
      3
      4
      5

NameError: name 's' is not defined
```

Next steps: Explain error

```python
data['orgyear']
```

```
0         2016.0
1         2018.0
2         2015.0
3         2017.0
4         2017.0
           ...
205838    2008.0
205839    2017.0
205840    2021.0
205841    2019.0
205842    2014.0
Name: orgyear, Length: 205843, dtype: float64
```

```python
data['job_position']=data['job_position'].fillna(data['job_position'].mode()[0])
```

```python
# creating class
```

```python
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      205843 non-null  object
 6   ctc_updated_year  205843 non-null  float64
dtypes: float64(2), int64(2), object(3)
memory usage: 11.0+ MB
```

```python
data.describe()
```

	Unnamed: 0	orgyear	ctc	ctc_updated_year
count	205843.000000	205757.000000	2.058430e+05	205843.000000
mean	103273.941786	2014.882750	2.271685e+06	2019.628231
std	59741.306484	63.571115	1.180091e+07	1.325104
min	0.000000	0.000000	2.000000e+00	2015.000000
25%	51518.500000	2013.000000	5.300000e+05	2019.000000
50%	103151.000000	2016.000000	9.500000e+05	2020.000000
75%	154992.500000	2018.000000	1.700000e+06	2021.000000
max	206922.000000	20165.000000	1.000150e+09	2021.000000

```python
data.duplicated()
```

```
0         False
1         False
2         False
3         False
4         False
          ...
205838    False
205839    False
205840    False
205841    False
205842    False
Length: 205843, dtype: bool
```

```python
data.head(5)
```

	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

```
# manual clustering
# creating designation flag and insights
data['job_position'].unique()
```

```
array(['Other', 'FullStack Engineer', 'Backend Engineer', ...,
       'Web / UI Designer', 'Azure data Factory',
       'Android Application developer'], dtype=object)
```

```
data['job_position']=data['job_position'].map({'Other':0,'FullStack Engineer':1,'FullStack Engineer':2,'Backend Engineer':3,'Web / UI Designer':4,'Azure data Factory':5,
                                              'Android Application developer':6,})
```

```
data.groupby('job_position')['ctc'].mean()
# thes are average ctc values
```

```
job_position
0.0    3.973584e+06
2.0    1.871618e+06
3.0    1.983635e+06
4.0    1.019999e+06
5.0    6.700000e+05
6.0    1.500000e+06
Name: ctc, dtype: float64
```

```
data['ctc'].unique()
```

```
array([1100000,  449999, 2000000, ..., 5266000,  234000, 3327000])
```

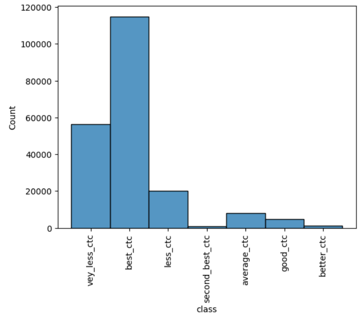
```
data1=data.copy()
```

```
# creating class flag and insights
def lk(ctc):
  if ctc>1000000 and ctc<=2000000 :
    return 'vey_less_ctc'
  elif ctc>2000000 and ctc<=3000000:
    return 'less_ctc'
  elif ctc>3000000 and ctc<=4000000:
    return 'average_ctc'
  elif ctc>4000000 and ctc<=6000000:
    return 'good_ctc'
  elif ctc>6000000 and ctc<=8000000:
    return 'better_ctc'
  elif ctc>8000000 and ctc<=10000000:
    return 'second_best_ctc'
  else:
    return 'best_ctc'
```

```
data['class']=data['ctc'].apply(lk)
import seaborn as sns
```

```
sns.histplot(x=data['class'],data=data)
import matplotlib.pyplot as plt
plt.xticks(rotation=90)
```

```
([0, 1, 2, 3, 4, 5, 6],
 [Text(0, 0, 'vey_less_ctc'),
  Text(1, 0, 'best_ctc'),
  Text(2, 0, 'less_ctc'),
  Text(3, 0, 'second_best_ctc'),
  Text(4, 0, 'average_ctc'),
  Text(5, 0, 'good_ctc'),
  Text(6, 0, 'better_ctc')])
```




```
# here best ctc is more
```

```
# creating tier  flag  and insights
def pk(ctc):
  if ctc>1000000 and ctc<=2000000 :
    return 'A'
  elif ctc>2000000 and ctc<=3000000:
    return 'B'
  elif ctc>3000000 and ctc<=4000000:
    return 'c'
  elif ctc>4000000 and ctc<=6000000:
    return 'C'
  elif ctc>6000000 and ctc<=8000000:
    return 'D'
  elif ctc>8000000 and ctc<=10000000:
    return 'E'
  else:
    return 'F'
```

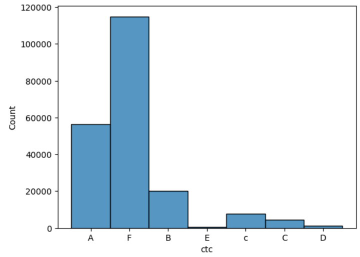
```
data['ctc']=data['ctc'].apply(pk)
```

```
l=data['ctc'].value_counts()
l
```

```
ctc
F    114912
A     56489
B     20132
c      7904
C      4621
D      1086
E       699
Name: count, dtype: int64
```

```
import seaborn as sns
sns.histplot(x=data['ctc'])
# the frequency of occurence of f category is more means more people are having ctc  greater than 10000000
```

```
<Axes: xlabel='ctc', ylabel='Count'>
```




```
from sklearn.preprocessing import LabelEncoder
```

Start coding or generate with AI.

```
lr=LabelEncoder()
```

```
data1['company_hash']=lr.fit_transform(data[['company_hash']])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_label.py:116: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
data1['email_hash']=lr.fit_transform(data['email_hash'])
```

```
data1['job_position']=lr.fit_transform(data['job_position'])
```

```
data1
```

	Unnamed: 0	company_hash	email_hash	orgyear	ctc	job_position	ctc_updated_year
0	0	969	65787	2016.0	1100000	0	2020.0
1	1	19729	105915	2018.0	449999	1	2019.0
2	2	15511	43304	2015.0	2000000	2	2020.0
3	3	12107	143869	2017.0	700000	2	2019.0
4	4	20225	66994	2017.0	1400000	1	2019.0
...	...	...	...	...	...	...	...
205838	206918	28751	67036	2008.0	220000	2	2019.0
205839	206919	8508	76247	2017.0	500000	2	2020.0
205840	206920	29084	121664	2021.0	700000	2	2021.0
205841	206921	36030	150680	2019.0	5100000	2	2019.0
205842	206922	2167	7283	2014.0	1240000	2	2016.0

205843 rows × 7 columns

```
# feature scaling
from sklearn.preprocessing import MinMaxScaler
```

```
m=MinMaxScaler()
import pandas as pd
```

```
data1=pd.DataFrame(m.fit_transform(data1),columns=data1.columns)
```

```
data1.isna().sum()
data1['orgyear']=data1['orgyear'].fillna(data1['orgyear'].mode()[0])
data1.isna().sum()
```

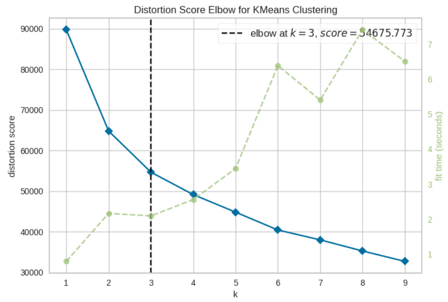
```
Unnamed: 0          0
company_hash        0
email_hash          0
orgyear             0
ctc                 0
job_position        0
ctc_updated_year    0
dtype: int64
```

```
# unsupervised learning
# elbow method
g={}
from sklearn.cluster import KMeans
for i in range(1,11):
  model=KMeans(n_clusters=i)
  c=model.fit(data1)
  g[i]=c.inertia_
```

```
/usr/local/lib/python3.10/dist-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(
/usr/local/lib/python3.10/dist-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(
/usr/local/lib/python3.10/dist-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(
/usr/local/lib/python3.10/dist-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(
/usr/local/lib/python3.10/dist-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(
/usr/local/lib/python3.10/dist-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(
/usr/local/lib/python3.10/dist-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(
/usr/local/lib/python3.10/dist-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(
/usr/local/lib/python3.10/dist-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(
/usr/local/lib/python3.10/dist-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(
```

```
from yellowbrick.cluster import KElbowVisualizer
vis=KElbowVisualizer(KMeans(),k=(1,10))
vi=vis.fit(data1)
vi.show()
```

```
/usr/local/lib/python3.10/dist-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(
/usr/local/lib/python3.10/dist-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(
/usr/local/lib/python3.10/dist-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(
/usr/local/lib/python3.10/dist-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(
/usr/local/lib/python3.10/dist-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(
/usr/local/lib/python3.10/dist-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(
/usr/local/lib/python3.10/dist-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(
/usr/local/lib/python3.10/dist-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(
/usr/local/lib/python3.10/dist-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(
```




```
<Axes: title={'center': 'Distortion Score Elbow for KMeans Clustering'}, xlabel='k', ylabel='distortion score'>
```

```
#3 clusters are more desirable
# hierarchial clustering
data1
```

	Unnamed: 0	company_hash	email_hash	orgyear	ctc	job_position	ctc_updated_year
0	0.000000	0.025979	0.428742	0.099975	0.001100	0.000000	0.833333
1	0.000005	0.528942	0.690261	0.100074	0.000450	0.166667	0.666667
2	0.000010	0.415856	0.282217	0.099926	0.002000	0.333333	0.833333
3	0.000014	0.324593	0.937612	0.100025	0.000700	0.333333	0.666667
4	0.000019	0.542240	0.436608	0.100025	0.001400	0.166667	0.666667
...	...	...	...	...	...	...	...
205838	0.999981	0.770825	0.436882	0.099578	0.000220	0.333333	0.666667
205839	0.999986	0.228103	0.496911	0.100025	0.000500	0.333333	0.833333
205840	0.999990	0.779753	0.792899	0.100223	0.000700	0.333333	1.000000
205841	0.999995	0.965978	0.982000	0.100124	0.005099	0.333333	0.666667
205842	1.000000	0.058098	0.047464	0.099876	0.001240	0.333333	0.166667

205843 rows × 7 columns

```
from sklearn.cluster import AgglomerativeClustering
```

```
ag=AgglomerativeClustering(n_clusters=3)
```

```
p=ag.fit(data1.head(1500))
```

```
p
```

```
AgglomerativeClustering
AgglomerativeClustering(n_clusters=3)
```

```
#Business Insights:
#Talent Profiling: Gain insights into the types of learners attracted to Scalers courses based on their job profiles and company affiliations.
#Company Preferences: Identify which companies employees are more inclined towards upskilling through Scaler's courses.
#Course Effectiveness: Evaluate the effectiveness of Scaler's courses by examining learner performance within different clusters.
#Market Segmentation: Segment the market of potential learners based on their characteristics, allowing for targeted marketing strategies.


# Recommendations:
#Tailored Course Offerings: Develop specialized courses or modules tailored to the needs and preferences of specific learner clusters.
#Strategic Partnerships: Forge partnerships with companies whose employees exhibit a high propensity for upskilling through Scaler.
#Personalized Learning Paths: Offer personalized learning paths or recommendations based on the learner cluster they belong to, maximizing enga
#Continuous Improvement: Continuously refine course content and delivery based on insights gained from clustering analysis to ensure relevance
```