Purpose:

fhfhs;fh['ifhoishfoisdfhjbjfpjufpsifspfjspofjsoifhjsdoijoivhnjsaovh;ovhv fhojf'sj'fj'F JOPJ'FPOJ'FJ'fpjF KJOJFCP'fcj'PFJP'afcj fioi;fjoifAOFJ

Approach:

FNJ;HF;IOFHWFHWOE;IHJFOWJF;OJ WFCNWOIHF;OIWFHJCWHFC J;OIJH;IOWCVFOIWHOJHWVCFH HHFC;WOHVCFOWHVOIVHOIWHCVFOWVHOIWV CVJOIVFCHIVHIO WV;OIHVJ;OIVOJHV WHIFW;OFWOIFHOW;NVVOJHV;H VCOWIJV;IOVCJ;CVFNJOI

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
file_path = '/content/drive/My Drive/scaler_clustering.csv'
```

```
# Import necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import re
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.impute import KNNImputer
from scipy.cluster.hierarchy import dendrogram, linkage
```

```
# Load the dataset
df = pd.read_csv(file_path)
```

```
# Check the first few rows of the dataset
print(df.head())
```

```
   index               company_hash  \
0      0             atrgxnnt xzaxv   
1      1  qtrxvzwt xzegwgbb rxbxnta   
2      2              ojzwnvwnxw vx   
3      3                  ngpgutaxv   
4      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
1  FullStack Engineer              2019
2    Backend Engineer              2020
3    Backend Engineer              2019
4  FullStack Engineer              2019
```

```
# Checking the shape, data types, and missing values
print(df.shape)
```

```
(205843, 7)
```

```
# Check for duplicates
duplicates = df.duplicated()
```

```
# Count the number of duplicate rows
num_duplicates = duplicates.sum()
print(f"Number of duplicate rows: {num_duplicates}")
```

```
Number of duplicate rows: 0
```

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205843 entries, 0 to 205842
Data columns (total 7 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   index             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  int64  
dtypes: float64(1), int64(3), object(3)
memory usage: 11.0+ MB
None
```

I see that orgyear (which is "employment start year",) has afew values which are future years and a few values which lead to more than 50 years of experience, which is not plausible. I would like to remove these rows from the dataset.

```
# Set the current year
current_year = 2024
```

```
# Detect and eliminate rows where 'orgyear' is in the future or leads to more than 50 years of experience
# Define a valid range for 'orgyear': should be less than or equal to the current year, and not more than 50 years old
valid_orgyear = (df['orgyear'] <= current_year) & (df['orgyear'] >= current_year - 50)
```

```
# Filter the dataset to keep only the valid records
df_cleaned = df[valid_orgyear]
```

```
# Check the shape of the cleaned dataframe and number of rows removed
print(f"Original dataset size: {df.shape}")
print(f"Cleaned dataset size: {df_cleaned.shape}")
print(f"Number of invalid records removed: {df.shape[0] - df_cleaned.shape[0]}")
```

```
Original dataset size: (205843, 7)
Cleaned dataset size: (205665, 7)
Number of invalid records removed: 178
```

```
# Get the invalid rows that were removed
invalid_rows = df[~valid_orgyear]
```

```
# Display the invalid rows
print(invalid_rows)
```

```
         index                  company_hash  \
2211      2211                        phrxkv   
2333      2333    xgmgn ntwyzgrgsxto ucn rna   
2562      2562                            tj   
3122      3122                      ft tdwtr   
3365      3365                   fyxntyvn lq   
...        ...                           ...   
196354  197352          vaxnjv mxqrv wvuxnvr   
198187  199212                 xb  v onhatzn   
202210  203276                     mqvmtzatq   
203992  205068  xatv ouvqp ogrhnxgzo ucn rna   
205435  206515                      vhngsqxa   

                                               email_hash  orgyear      ctc  \
2211    3394674bb6bb1de6289e931853fa0bd131c811e0054a92...   2031.0  1500000   
2333    c737ceb66c7f0ce37c2fce087003aa129632a3a2fa4f6c...      NaN   170000   
2562    25edac17c77f6f0edeafb86f7a7844d96dc899e193c87e...      NaN   860000   
3122    c402eba160abf4e5b5f72af775fc98dbd346f1a081112e...      NaN   600000   
3365    38bd913564fa983cd4fb7799e4027d8ed2b0fd6263e15a...      NaN   800000   
...                                                   ...      ...      ...   
196354  069308440811d578c817c05392f97e8919baac6aa12aa3...      1.0  2900000   
198187  9429a19771ae913f169917d380c94f003115aaaf904388...   2025.0   300000   
202210  d66f939c4318c1958be5bc9e7b70b741aa61be7493ff58...   2028.0  1300000   
203992  7191da2e57dcb0c1301711e889ea72d5cc801e039359b1...  20165.0   850000   
205435  3fa8de870da01d863abba8eb6a8ae3df1aa18c18946688...      NaN  2400000   

            job_position  ctc_updated_year  
2211    Backend Engineer              2020  
2333               Other              2020  
2562        Data Analyst              2020  
```

```
3122    Support Engineer              2020
3365                 NaN              2021
...                  ...               ...
196354    Data Scientist              2019
198187             Other              2021
202210  Backend Engineer              2021
203992               NaN              2019
205435               NaN              2020

[178 rows x 7 columns]
```

```
print(df_cleaned.isnull().sum())
```

```
index                   0
company_hash           44
email_hash              0
orgyear                 0
ctc                     0
job_position        52508
ctc_updated_year        0
dtype: int64
```

```
# Mode imputation for company_hash
df_cleaned['company_hash'].fillna(df_cleaned['company_hash'].mode()[0], inplace=True)
```

```
<ipython-input-17-f64e7f3aeabb>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_cleaned['company_hash'].fillna(df_cleaned['company_hash'].mode()[0], inplace=True)
```

```
print(df_cleaned.isnull().sum())
```

```
index                   0
company_hash            0
email_hash              0
orgyear                 0
ctc                     0
job_position        52508
ctc_updated_year        0
dtype: int64
```

```
# Fill missing values (Mean/ KNN Imputation)
# imputer = KNNImputer(n_neighbors=5)
```

```
# df_cleaned[['orgyear']] = imputer.fit_transform(df_cleaned[['orgyear']])
```

```
# print(df_cleaned.isnull().sum())
```

```
# Convert orgyear
df_cleaned['orgyear'] = df_cleaned['orgyear'].astype(int)
```

```
<ipython-input-22-40f87f736f36>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_cleaned['orgyear'] = df_cleaned['orgyear'].astype(int)
```

```
print(df_cleaned.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 205665 entries, 0 to 205842
Data columns (total 7 columns):
 #   Column            Non-Null Count   Dtype 
---  ------            --------------   ----- 
 0   index             205665 non-null  int64 
 1   company_hash      205665 non-null  object
 2   email_hash        205665 non-null  object
 3   orgyear           205665 non-null  int64 
 4   ctc               205665 non-null  int64 
 5   job_position      153157 non-null  object
 6   ctc_updated_year  205665 non-null  int64 
dtypes: int64(4), object(3)
memory usage: 12.6+ MB
None
```

```
print(df_cleaned.isnull().sum())
```

```
index                   0
company_hash            0
email_hash              0
orgyear                 0
ctc                     0
job_position        52508
ctc_updated_year        0
dtype: int64
```

```
# Group by company_hash and fill missing job_position with mode for that company
df_cleaned['job_position'] = df_cleaned.groupby('company_hash')['job_position'].transform(lambda x: x.fillna(x.mode()[0] if not x.mode().empty else 'Unknown'))
```

```
<ipython-input-25-52a7906e5793>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_cleaned['job_position'] = df_cleaned.groupby('company_hash')['job_position'].transform(lambda x: x.fillna(x.mode()[0] if not x.mode().empty else 'Unknown'))
```

```
print(df_cleaned.isnull().sum())
```

```
index               0
company_hash        0
email_hash          0
orgyear             0
ctc                 0
job_position        0
ctc_updated_year    0
```

```
dtype: int64
```

```
###############################################################################################
```

```
df_cleaned.head()
```

	index	company_hash	email_hash	orgyear	ctc	job_position	ctc_updated_year
0	0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016	1100000	Other	2020
1	1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018	449999	FullStack Engineer	2019
2	2	ojzwnvwnxw vx	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2015	2000000	Backend Engineer	2020
3	3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	2017	700000	Backend Engineer	2019
4	4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	2017	1400000	FullStack Engineer	2019

```
# Unique email hashes and frequency of occurrences
print(df_cleaned['email_hash'].value_counts())
```

```
email_hash
bbace3cc586400bbc65765bc6a16b77d8913836cfc98b77c05488f02f5714a4b    10
3e5e49daa5527a6d5a33599b238bf9bf31e85b9efa9a94f1c88c5e15a6f31378     9
6842660273f70e9aa239026ba33bfe82275d6ab0d20124021b952b5bc3d07e6c     9
298528ce3160cc761e4dc37a07337ee2e0589df251d73645aae209b010210eee     9
d15041f58bb01c8ee29f72e33b136e26bc32f3169a40b53d75fe7ae9cbb9a551     8
                                                                    ..
6ed7767a6ba36e8ab4f4d2397a4d32f26f34387720645906bf51a05c2152fd56     1
9778d2fa1bbfb721c3e90941cb3474740610d301f2ccf1429f5c6835ae5e27f4     1
9a891d279335db60cd6a45c2243bca2c56f940e31c5a812a6f642ea800832c4b     1
e96207e084f4552ba131598c704d2c5f12373999fc66285f58dea00afb9d333c     1
0bcfc1d05f2e8dc4147743a1313aa70a119b41b30d4a1f7e738a6a87d3712c31     1
Name: count, Length: 153292, dtype: int64
```

```
# Remove special characters from Company_hash
df_cleaned['company_hash_cleaned'] = df_cleaned['company_hash'].apply(lambda x: re.sub('[^A-Za-z0-9 ]+', '', str(x)))
```

```
<ipython-input-30-eb3fda435fdc>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_cleaned['company_hash_cleaned'] = df_cleaned['company_hash'].apply(lambda x: re.sub('[^A-Za-z0-9 ]+', '', str(x)))
```

```
# Unique "company_hash_cleaned" hashes and frequency of occurrences
print(df_cleaned['company_hash_cleaned'].value_counts())
```

```
company_hash_cleaned
nvnv wgzohrnvzwj otqcxwto    8379
xzegojo                      5378
vbvkgz                       3480
zgn vuurxwvmrt vwwghzn       3408
wgszxkvzn                    3238
                             ...
```

```
bvpt owyggr                   1
vhngsqxa xzaxv                1
ctavznh td kteg               1
ihxwprgsxw ogenfvqt           1
bvptbjnqxu td vbvkgz          1
Name: count, Length: 37246, dtype: int64
```

```
# Dropping the 'company_hash' column from the DataFrame
df_cleaned.drop(columns=['company_hash'], inplace=True)
```

```
<ipython-input-32-d2a89671a684>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_cleaned.drop(columns=['company_hash'], inplace=True)
```

```
print(df_cleaned.shape)
```

```
(205665, 7)
```

```
# Create new feature 'Years_of_Experience'
current_year = 2024
df_cleaned['Years_of_Experience'] = current_year - df_cleaned['orgyear']
```

```
<ipython-input-34-c3a4f380c3bf>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_cleaned['Years_of_Experience'] = current_year - df_cleaned['orgyear']
```

```
df_cleaned.head()
```

	index	email_hash	orgyear	ctc	job_position	ctc_updated_year	company_hash_cleaned	Years_of_Experience
0	0	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016	1100000	Other	2020	atrgxnnt xzaxv	8
1	1	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018	449999	FullStack Engineer	2019	qtrxvzwt xzegwgbb rxbxnta	6
2	2	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2015	2000000	Backend Engineer	2020	ojzwnvwnxw vx	9
3	3	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	2017	700000	Backend Engineer	2019	ngpgutaxv	7
4	4	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	2017	1400000	FullStack Engineer	2019	qxen sqghu	7

```
from google.colab import files
```

```
# # Export the cleaned DataFrame to Excel
# df_cleaned.to_excel('scaler_clustering_pre_pro_01.xlsx', index=False)
```

```
# # Download the file
# files.download('scaler_clustering_pre_pro_01.xlsx')
# Five-point summary of CTC
grouped = df_cleaned.groupby(['company_hash_cleaned', 'job_position', 'Years_of_Experience'])['ctc'].agg(['mean', 'median', 'max', 'min', 'count'])

# Merge it back to the original dataframe
df_cleaned = df_cleaned.merge(grouped, how='left', on=['company_hash_cleaned', 'job_position', 'Years_of_Experience'])

# Create flags for designation, class, and tier
df_cleaned['Designation'] = np.where(df_cleaned['ctc'] > df_cleaned['mean'], 1, 0)
df_cleaned['Class'] = pd.qcut(df_cleaned['ctc'], 3, labels=[3, 2, 1])
df_cleaned['Tier'] = pd.qcut(df_cleaned['ctc'], 3, labels=[3, 2, 1])
```

```
# Export the cleaned DataFrame to Excel
df_cleaned.to_excel('scaler_clustering_pre_pro_02.xlsx', index=False)
```

```
# Download the file
files.download('scaler_clustering_pre_pro_02.xlsx')
```

```
df1=df_cleaned.copy()  # used for downsizing for hierarchical clustering at later steps
```

```
df_cleaned.head()
```

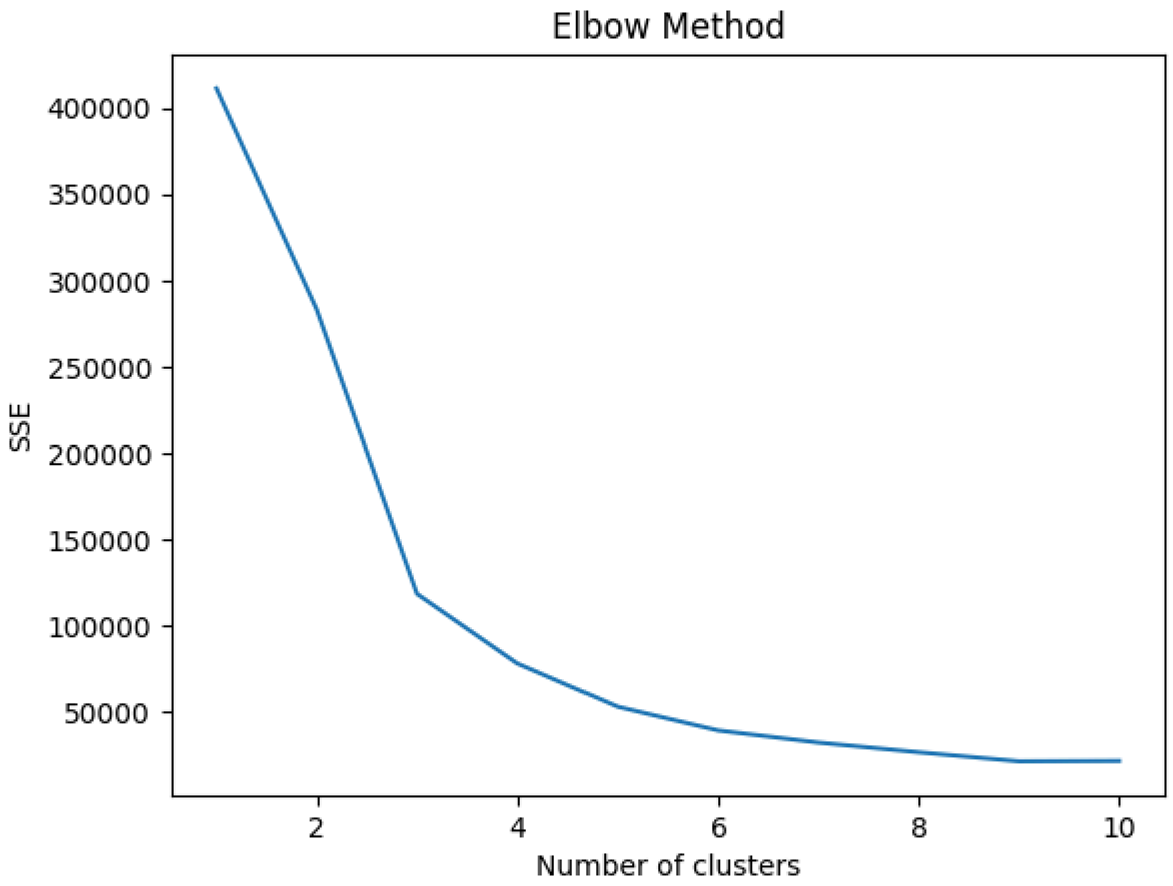
	index	email_hash	orgyear	ctc	job_position	ctc_updated_year	company_hash_cleaned	Years_of_Experience	mean	median	max	
0	0	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016	1100000	Other	2020	atrgxnnt xzaxv	8	1.100000e+06	1100000.0	1100000	110
1	1	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018	449999	FullStack Engineer	2019	qtrxvzwt xzegwgbb rxbxnta	6	7.742856e+05	750000.0	1200000	44
2	2	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2015	2000000	Backend Engineer	2020	ojzwnvwnxw vx	9	2.000000e+06	2000000.0	2000000	200
3	3	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	2017	700000	Backend Engineer	2019	ngpgutaxv	7	1.436154e+06	1210000.0	3160000	70
4	4	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	2017	1400000	FullStack Engineer	2019	qxen sqghu	7	1.400000e+06	1400000.0	1400000	140

Start coding or generate with AI.

```
# Standardize the numeric columns for clustering
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df_cleaned[['ctc', 'Years_of_Experience']])
```

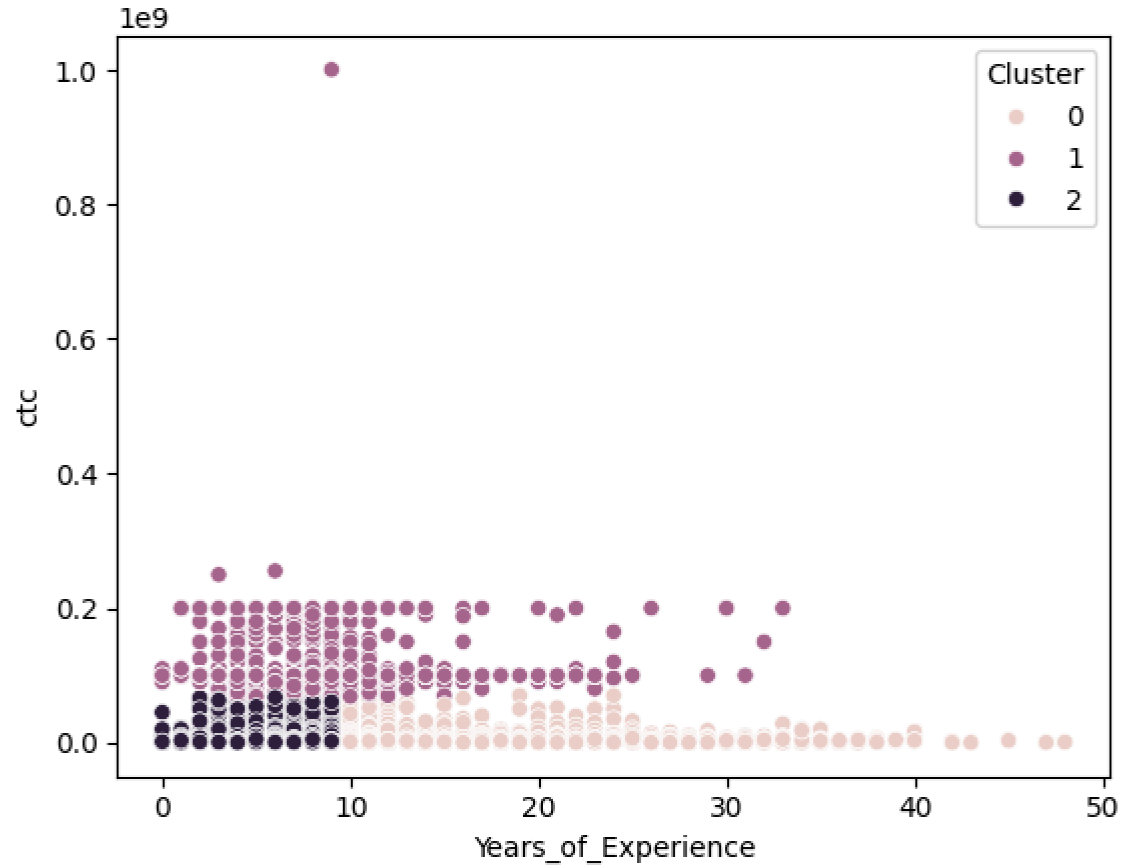
```
# Elbow method to find the optimal number of clusters
sse = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(df_scaled)
    sse.append(kmeans.inertia_)

# Plot the Elbow graph
plt.plot(range(1, 11), sse)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('SSE')
plt.show()
```



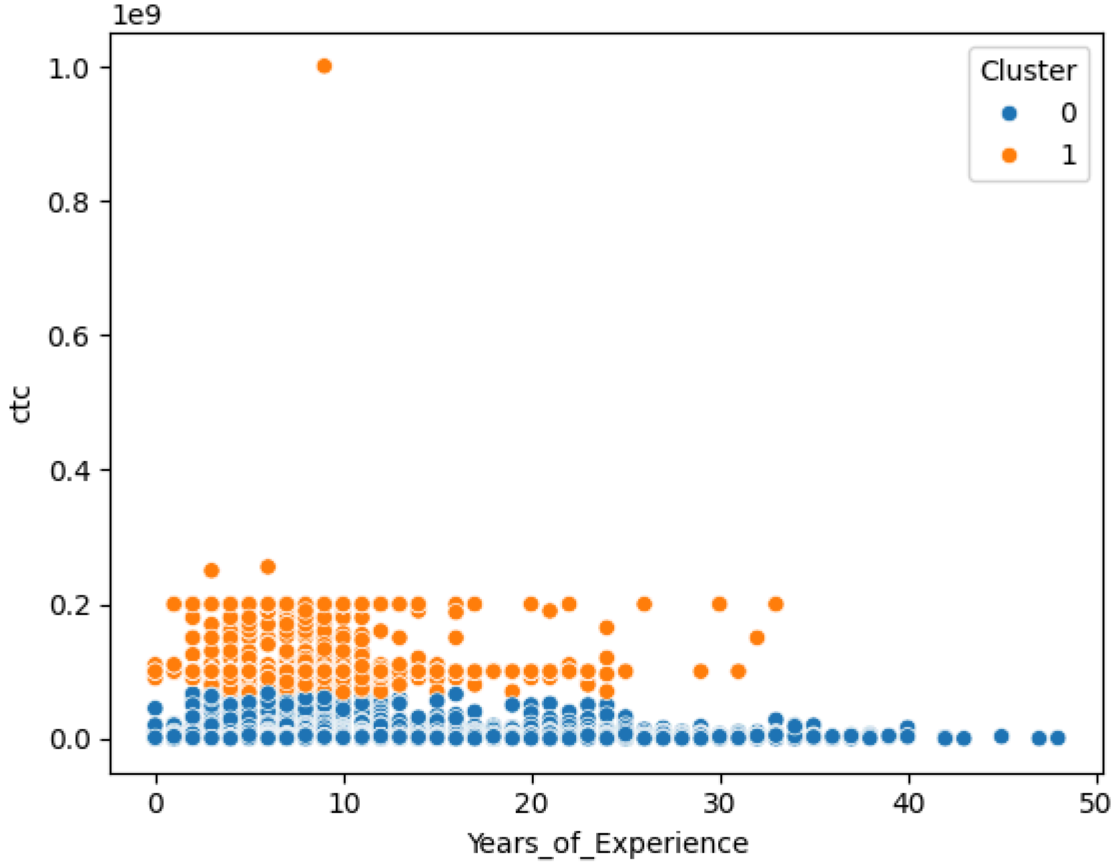
```
# Applying KMeans
kmeans = KMeans(n_clusters=3)
df_cleaned['Cluster'] = kmeans.fit_predict(df_scaled)

# Visualize the clusters
sns.scatterplot(x='Years_of_Experience', y='ctc', hue='Cluster', data=df_cleaned)
plt.show()
```



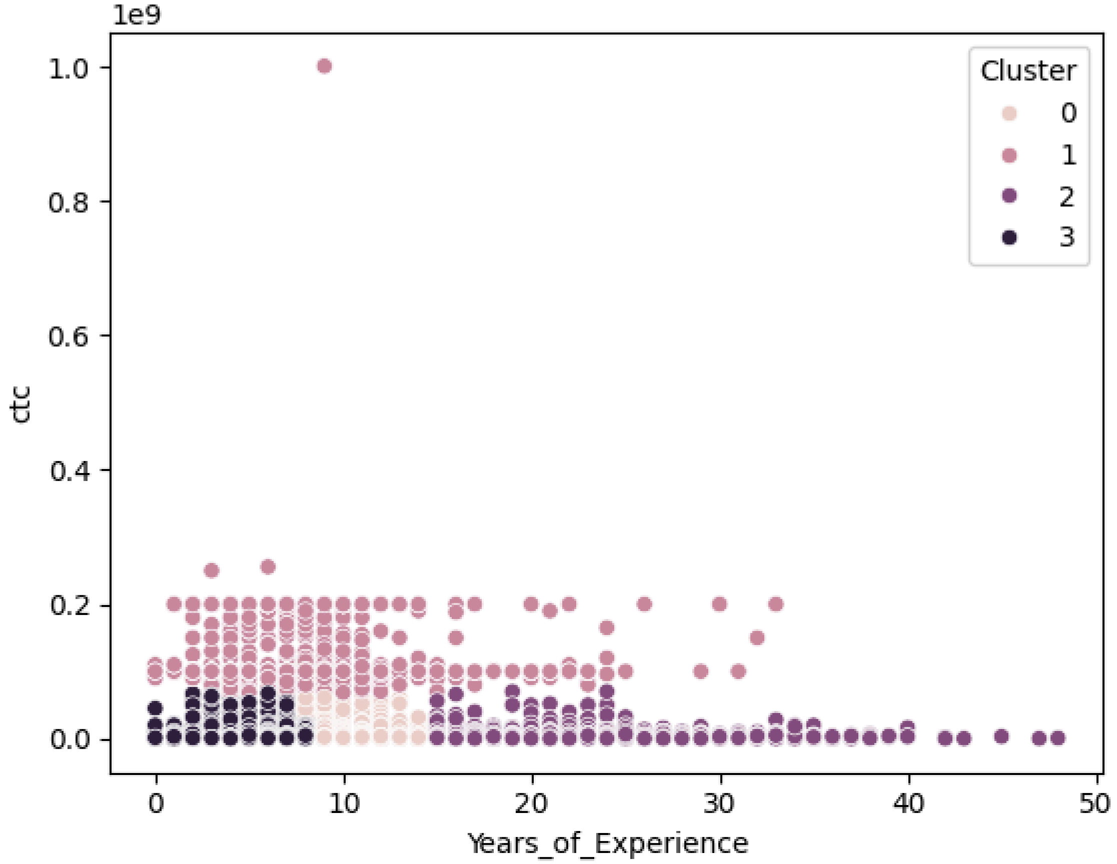
```
# Applying KMeans
kmeans = KMeans(n_clusters=2)
df_cleaned['Cluster'] = kmeans.fit_predict(df_scaled)

# Visualize the clusters
sns.scatterplot(x='Years_of_Experience', y='ctc', hue='Cluster', data=df_cleaned)
plt.show()
```



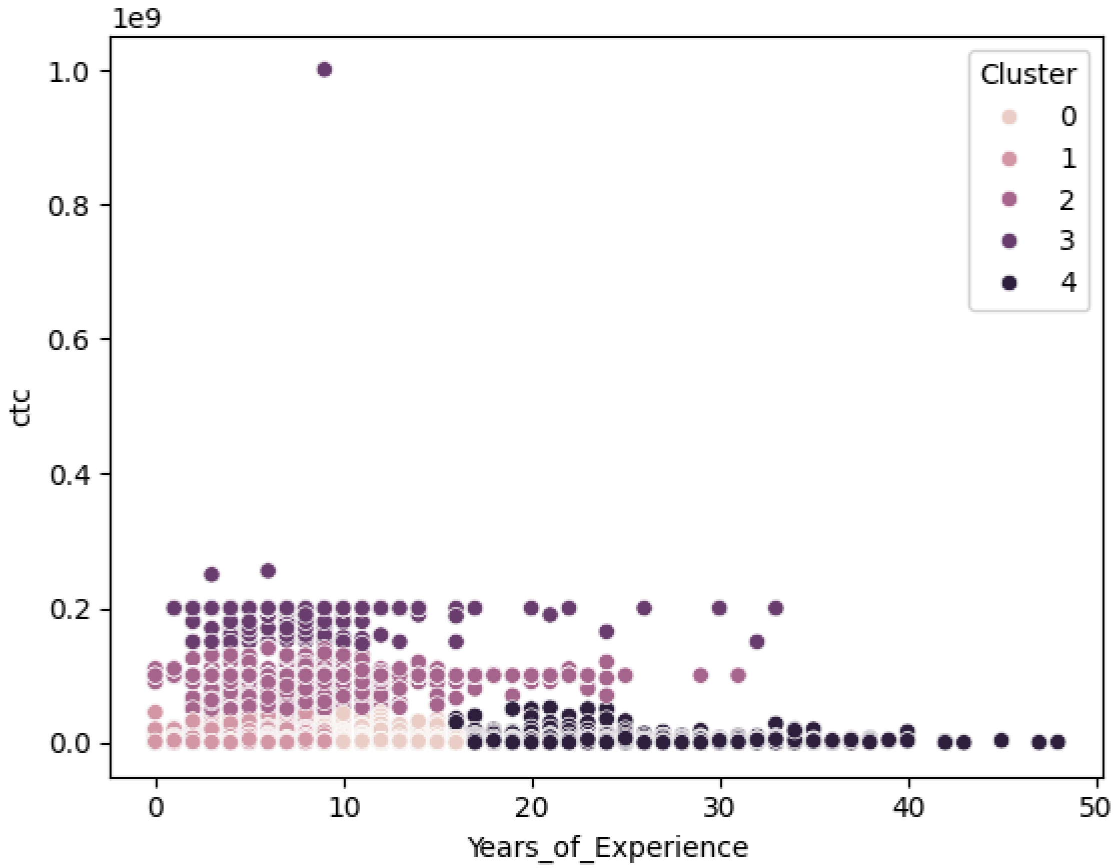
```
# Applying KMeans
kmeans = KMeans(n_clusters=4)
df_cleaned['Cluster'] = kmeans.fit_predict(df_scaled)

# Visualize the clusters
sns.scatterplot(x='Years_of_Experience', y='ctc', hue='Cluster', data=df_cleaned)
plt.show()
```



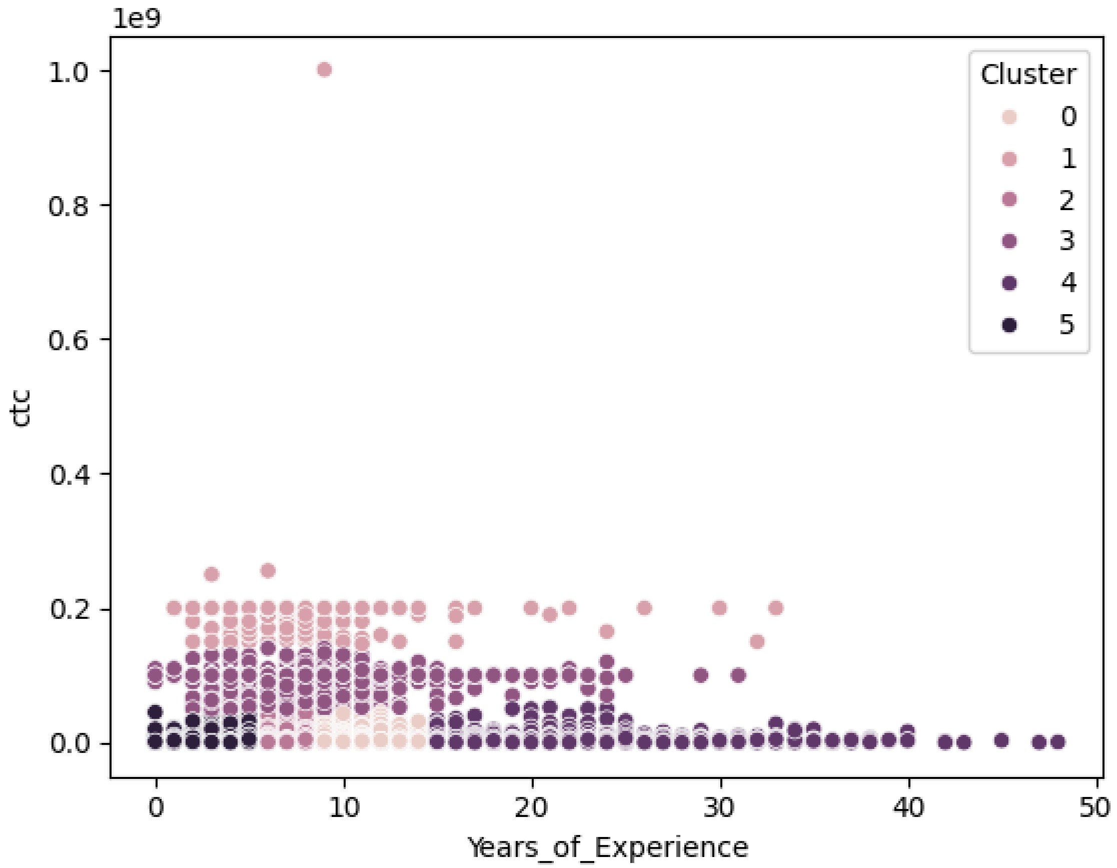
```
# Applying KMeans
kmeans = KMeans(n_clusters=5)
df_cleaned['Cluster'] = kmeans.fit_predict(df_scaled)

# Visualize the clusters
sns.scatterplot(x='Years_of_Experience', y='ctc', hue='Cluster', data=df_cleaned)
plt.show()
```



```
# Applying KMeans
kmeans = KMeans(n_clusters=6)
df_cleaned['Cluster'] = kmeans.fit_predict(df_scaled)

# Visualize the clusters
sns.scatterplot(x='Years_of_Experience', y='ctc', hue='Cluster', data=df_cleaned)
plt.show()
```



```
# Check the summary statistics of orgyear
print(df_cleaned['orgyear'].describe())
```

```
count    205665.000000
mean       2015.117584
std           4.228364
min        1976.000000
25%        2013.000000
50%        2016.000000
75%        2018.000000
max        2024.000000
Name: orgyear, dtype: float64
```

```
# Check for any unusual values (e.g., future years or very old years)
print(df_cleaned['orgyear'].value_counts().sort_index())
```

```
orgyear
1976        1
1977        1
1979        1
1981        1
1982        4
1984        3
1985        5
1986        8
1987        6
1988       10
1989       22
1990       38
1991       79
1992       47
```

```
1993       74
1994       65
1995       94
1996      134
1997      234
1998      279
1999      340
2000      495
2001      713
2002      685
2003     1018
2004     1455
2005     1873
2006     2075
2007     2257
2008     2728
2009     3777
2010     5751
2011     7970
2012    10493
2013    12351
2014    16696
2015    20610
2016    23043
2017    23239
2018    25256
2019    23427
2020    13431
2021     3670
2022      911
2023      252
2024       43
Name: count, dtype: int64
```

```
# Check the summary statistics of orgyear
print(df_cleaned['Years_of_Experience'].describe())
```

```
count    205665.000000
mean          8.882416
std           4.228364
min           0.000000
25%           6.000000
50%           8.000000
75%          11.000000
max          48.000000
Name: Years_of_Experience, dtype: float64
```

```
# Check for any unusual values (e.g., future years or very old years)
print(df_cleaned['Years_of_Experience'].value_counts().sort_index())
```

```
Years_of_Experience
0        43
1       252
2       911
3      3670
4     13431
5     23427
6     25256
```

```
7     23239
8     23043
9     20610
10    16696
11    12351
12    10493
13     7970
14     5751
15     3777
16     2728
17     2257
18     2075
19     1873
20     1455
21     1018
22      685
23      713
24      495
25      340
26      279
27      234
28      134
29       94
30       65
31       74
32       47
33       79
34       38
35       22
36       10
37        6
38        8
39        5
40        3
42        4
43        1
45        1
47        1
48        1
Name: count, dtype: int64
```

```
# # Perform hierarchical clustering
# Z = linkage(df_scaled, 'ward')
# dendrogram(Z)
# plt.show()
```

Start coding or generate with AI.

```
type(df1)
```