SCALER: CLUSTERING

Importing Libraries

In [454…

```
import pandas as pd
import numpy as np
import seaborn as sns
sns.set(style='whitegrid')
from scipy import stats
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from sklearn.preprocessing import MinMaxScaler
```

In [ ]:

Uploading the Dataset

In [455…

```
df = pd.read_csv("scaler_clustering.csv")
```

In [456…

```
df.head()
```

Out[456]:

	Unnamed: 0	company_hash	email_hash	orgyear	ct
0	0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	110000
1	1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018.0	44999
2	2	ojzwnvwnxw vx	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2015.0	200000
3	3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	2017.0	70000
4	4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	2017.0	140000

In [ ]:

Comments

We have 205843 data points and 7 features. We can drop the column Unnamed 0 as its the row Serial No

Also, our objective is clustering, the email_hash wont be useful feature as we wont be looking at the granularity of the data, but more focused on grouping the data into similar clusters. Therefore dropping email_hash as well.

Statistical Summary

In [457…
```
df.shape
```

Out[457]:
```
(205843, 7)
```

In [458…
```
# Creating a copy of original dataframe

df_org = df.copy()
```

In [459…
```
## Dropping 'Unnamed:0' and 'email_hash' column
```

In [460…
```
df.drop(columns=["Unnamed: 0","email_hash"], inplace = True)
```

In [461…
```
df.head()
```

Out[461]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year
0	atrgxnnt xzaxv	2016.0	1100000	Other	2020.0
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	FullStack Engineer	2019.0
2	ojzwnvwnxw vx	2015.0	2000000	Backend Engineer	2020.0
3	ngpgutaxv	2017.0	700000	Backend Engineer	2019.0
4	qxen sqghu	2017.0	1400000	FullStack Engineer	2019.0

In [ ]:
```
```

In [462…
```
## Missing Values
```

In [463…
```
def missingValue(df):
    #Identifying Missing data.
    total_null = df.isnull().sum().sort_values(ascending = False)
    percent = ((df.isnull().sum()/len(df))*100).sort_values(ascending = False)
    print(f"Total records in our data =  {df.shape[0]} where missing values are as

    missing_data = pd.concat([total_null,percent.round(2)],axis=1,keys=['Total Miss
    return missing_data
```

In [464…
```
missing_df = missingValue(df)
missing_df[missing_df['Total Missing'] > 0]
```

```
Total records in our data =  205843 where missing values are as follows:
```

Out[464]:

	Total Missing	In Percent
job_position	52562	25.53
orgyear	86	0.04
company_hash	44	0.02

Comments:

Total 3 features has missing values (job, year, company)

In [ ]:

Data Preprocessing

In [465…
```
# Using a regex function for removing special characters

import re
def remove_special (string):
    new_string=re.sub('[^A-Za-z ]+', '', string)
    return new_string
```

In [466…
```
#what happens here

mystring='\tAirtel\\\\&&**() X Labs'
re.sub('[^A-Za-z ]+', '', mystring)
```

Out[466]:
```
'Airtel X Labs'
```

In [467…
```
# Data Cleaning on job_position

df.job_position=df.job_position.apply(lambda x: remove_special(str(x)))
df.job_position=df.job_position.apply(lambda x: x.lower())
df.job_position=df.job_position.apply(lambda x: x.strip())
df.job_position
```

Out[467]:
```
0                       other
1          fullstack engineer
2            backend engineer
3            backend engineer
4          fullstack engineer
                  ...
205838                    nan
205839                    nan
205840                    nan
205841                    nan
205842                    nan
Name: job_position, Length: 205843, dtype: object
```

In [468…
```
df.shape
```

Out[468]:
```
(205843, 5)
```

In [469…
```
df.drop_duplicates(inplace=True)
df.shape
```

Out[469]:
```
(188247, 5)
```

In [470…
```
df['company_hash'].value_counts().sort_index()
```

Out[470]:
```
0                                  2
0000                               1
01 ojztqsj                         2
05mz exzytvrny uqxcvnt rxbxnta     2
1                                  2
                                  ..
zyvzwt wgzohrnxzs tzsxzttqo        1
zz                                 2
zzb ztdnstz vacxogqj ucn rna       2
zzgato                             1
zzzbzb                             1
Name: company_hash, Length: 37299, dtype: int64
```

In [ ]:
```
```

In [471…
```
# Data Cleaning on company_hash

df.company_hash=df.company_hash.apply(lambda x: remove_special(str(x)))
df.company_hash=df.company_hash.apply(lambda x: x.lower())
df.company_hash=df.company_hash.apply(lambda x: x.strip())
df.company_hash
```

Out[471]:
```
0                     atrgxnnt xzaxv
1          qtrxvzwt xzegwgbb rxbxnta
2                      ojzwnvwnxw vx
3                          ngpgutaxv
4                         qxen sqghu
                     ...
205838                     vuurt xzw
205839                     husqvawgb
205840                      vwwgrxnt
205841                zgn vuurxwvmrt
205842                bgqsvz onvzrtj
Name: company_hash, Length: 188247, dtype: object
```

In [472…
```
df['company_hash'].value_counts().sort_index()
```

Out[472]:
```
                                              85
a                                              1
a b onttr wgqu                                 1
a j uvnxr owyggr ge tzsxzttqxzs vwvatbj vbmx   1
a ntwy ogrhnxgzo ucn rna                       2
                                              ..
zz                                             2
zz wgzztwn mya                                 1
zzb ztdnstz vacxogqj ucn rna                   2
zzgato                                         1
zzzbzb                                         1
Name: company_hash, Length: 37208, dtype: int64
```

In [473…
```
print(df.shape)
print(df.drop_duplicates().shape)
df.drop_duplicates(inplace=True)
```

```
(188247, 5)
(188246, 5)
```

In [474…
```
#removing rows where company or job_position is not available

df=df[ ~((df['company_hash']=='') | (df['job_position']==''))]
```

In [475…
```
df.shape
```

Out[475]:
```
(188153, 5)
```

In [476…
```
df['orgyear'].isnull().sum()
```

Out[476]:
```
86
```

In [477…
```
company_median_org_year=df.groupby('company_hash')['orgyear'].median()
company_median_org_year
```

Out[477]:
```
company_hash
a                                              2017.0
a b onttr wgqu                                 2019.0
a j uvnxr owyggr ge tzsxzttqxzs vwvatbj vbmx    2015.0
a ntwy ogrhnxgzo ucn rna                       2013.0
a ntwyzgrgsxto                                 2015.0
                                                ...
zz                                             2011.0
zz wgzztwn mya                                 2009.0
zzb ztdnstz vacxogqj ucn rna                   2017.0
zzgato                                         2014.0
zzzbzb                                         1990.0
Name: orgyear, Length: 37205, dtype: float64
```

In [478…
```
#Code to impute

def null_imputation(table_from_which_we_need_to_fill, main_col, null_col):
    if np.isnan(null_col):
        return table_from_which_we_need_to_fill[main_col]
    else:
        return null_col
```

In [479…
```
# Filling Null values using Median Target Imputation for Orgyear

df['orgyear']=df.apply(lambda x: null_imputation(company_median_org_year,x['company
df['orgyear']
```

Out[479]:
```
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: 188153, dtype: float64
```

In [480…
```
#if we still have null values, we'll drop it

len(df[df['orgyear'].isnull()])
```

Out[480]:
```
26
```

In [481…
```
#dropping remaining null values

df=df[~df['orgyear'].isnull()]
```

In [482…
```
missing_df = missingValue(df)
missing_df[missing_df['Total Missing'] > 0]
```

```
Total records in our data =  188127 where missing values are as follows:
```

Out[482]:

	Total Missing	In Percent

In [ ]:

Outlier Detection and Treatment

- orgyear
- ctc

In [483…

```
#simple understanding

df.orgyear.value_counts().sort_values(ascending=True)
```

Out[483]:

```
200.0         1
208.0         1
38.0          1
2204.0        1
1900.0        1
           ...
2019.0    18551
2015.0    19613
2017.0    21320
2016.0    21477
2018.0    22157
Name: orgyear, Length: 79, dtype: int64
```

In [484…

```
#simple understanding

df.ctc.value_counts().sort_values(ascending=True)
```

Out[484]:

```
3327000        1
3102000        1
3149000        1
64429999       1
1849000        1
             ...
1200000     5623
500000      5662
800000      5917
1000000     6835
600000      6857
Name: ctc, Length: 3360, dtype: int64
```

In [485…

```
#removing outliers from orgyear using IQR

q1=df.orgyear.quantile(0.25)
q3=df.orgyear.quantile(0.75)
iqr=q3-q1
df=df.loc[(df.orgyear>=q1-1.5*iqr) & (df.orgyear<=q3+1.5*iqr)]

#removing outliers from ctc using IQR

q1=df.ctc.quantile(0.25)
q3=df.ctc.quantile(0.75)
iqr=q3-q1
df=df.loc[(df.ctc>=q1-1.5*iqr) & (df.ctc<=q3+1.5*iqr)]
```

In [486…

```
df.orgyear.value_counts().sort_index(ascending=True)
```

Out[486]:
```
2006.0    1635
2007.0    1821
2008.0    2279
2009.0    3215
2010.0    5004
          ...
2021.0    2900
2022.0     739
2023.0     200
2024.0      32
2025.0      11
Name: orgyear, Length: 22, dtype: int64
```

In [487…
```
print(df.shape)
print(df.drop_duplicates().shape)
df.drop_duplicates(inplace=True)
```

```
(168987, 5)
(168986, 5)
```

In [488…
```
df
```

Out[488]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0
2	ojzwnvwnxw vx	2015.0	2000000	backend engineer	2020.0
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0
...	...	...	...	...	...
205836	mvqwrvjo	2011.0	2250000	nan	2019.0
205838	vuurt xzw	2008.0	220000	nan	2019.0
205839	husqvawgb	2017.0	500000	nan	2020.0
205840	vwwgrxnt	2021.0	700000	nan	2021.0
205842	bgqsvz onvzrtj	2014.0	1240000	nan	2016.0

168986 rows × 5 columns

In [489…
```
#We see some 'nan's in job_position

df.loc[df['job_position']=='nan','job_position']=np.nan
```

In [490…
```
df
```

Out[490]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0
2	ojzwnvwnxw vx	2015.0	2000000	backend engineer	2020.0
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0
...	...	...	...	...	...
205836	mvqwrvjo	2011.0	2250000	NaN	2019.0
205838	vuurt xzw	2008.0	220000	NaN	2019.0
205839	husqvawgb	2017.0	500000	NaN	2020.0
205840	vwwgrxnt	2021.0	700000	NaN	2021.0
205842	bgqsvz onvzrtj	2014.0	1240000	NaN	2016.0

168986 rows × 5 columns

In [ ]:

In [491…
```
def feature_names(df):

    print(f"Columns with category datatypes (Categorical Features) are : \
    {list(df.select_dtypes('object').columns)}")
    print('-'*125)
    print('-'*125)
    print(f"Columns with integer and float datatypes (Numerical Features) are: \
    {list(df.select_dtypes(['int64','float64']).columns)}")
```

In [492…
```
feature_names(df)
```

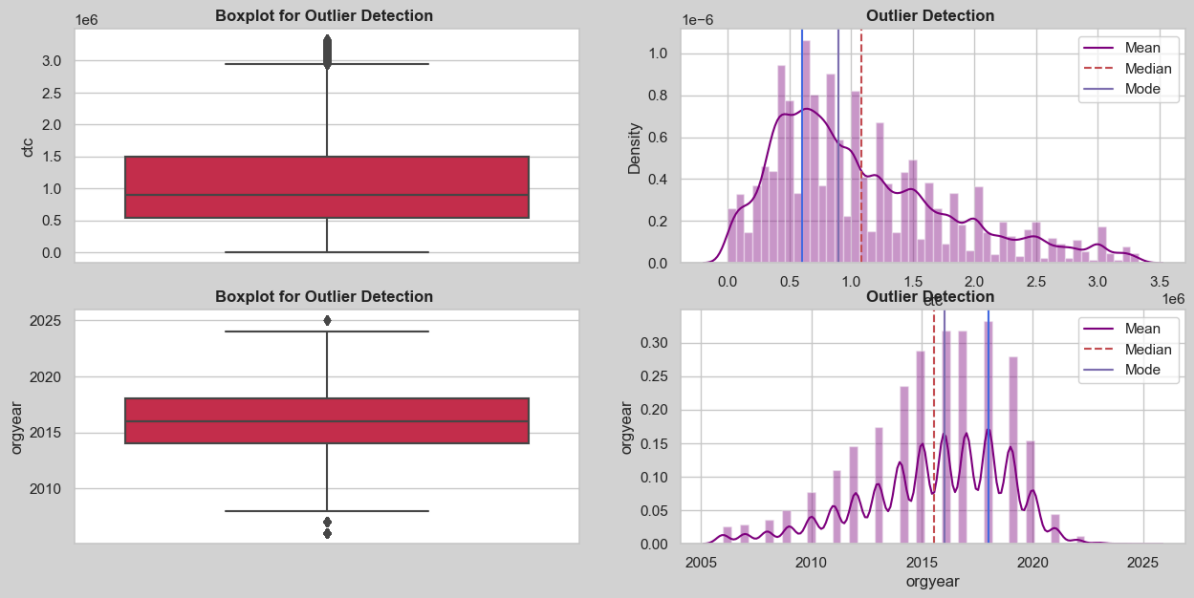
```
Columns with category datatypes (Categorical Features) are :     ['company_hash',
'job_position']
-----------------------------------------------------------------------------------
-----------------------------------------
-----------------------------------------------------------------------------------
-----------------------------------------
Columns with integer and float datatypes (Numerical Features) are:     ['orgyear',
'ctc', 'ctc_updated_year']
```

In [493…
```
def numerical_feat(df,colname,nrows=2,mcols=2,width=15,height=15):
    fig , ax = plt.subplots(nrows,mcols,figsize=(width,height))
    fig.set_facecolor("lightgrey")
    rows = 0
    for var in colname:
        ax[rows][0].set_title("Boxplot for Outlier Detection ", fontweight="bold")
        plt.ylabel(var, fontsize=12)
        sns.boxplot(y = df[var],color='crimson',ax=ax[rows][0])

        # plt.subplot(nrows,mcols,pltcounter+1)
        sns.distplot(df[var],color='purple',ax=ax[rows][1])
        ax[rows][1].axvline(df[var].mean(), color='r', linestyle='--', label="Mean'
        ax[rows][1].axvline(df[var].median(), color='m', linestyle='-', label="Medi
        ax[rows][1].axvline(df[var].mode()[0], color='royalblue', linestyle='-', la
        ax[rows][1].set_title("Outlier Detection ", fontweight="bold")
        ax[rows][1].legend({'Mean':df[var].mean(),'Median':df[var].median(),'Mode':
```

```
        rows += 1
    plt.show()
```

In [494…
```
# We won't consider 'ctc_updated_year' as numerical but instead categorical feature

numerical_cols = ['ctc', 'orgyear']
```

In [495…
```
numerical_feat(df,numerical_cols,len(numerical_cols),2,15,7)
```



In [ ]:

In [496…
```
categorical_cols = ['company_hash', 'job_position', 'ctc_updated_year']
```

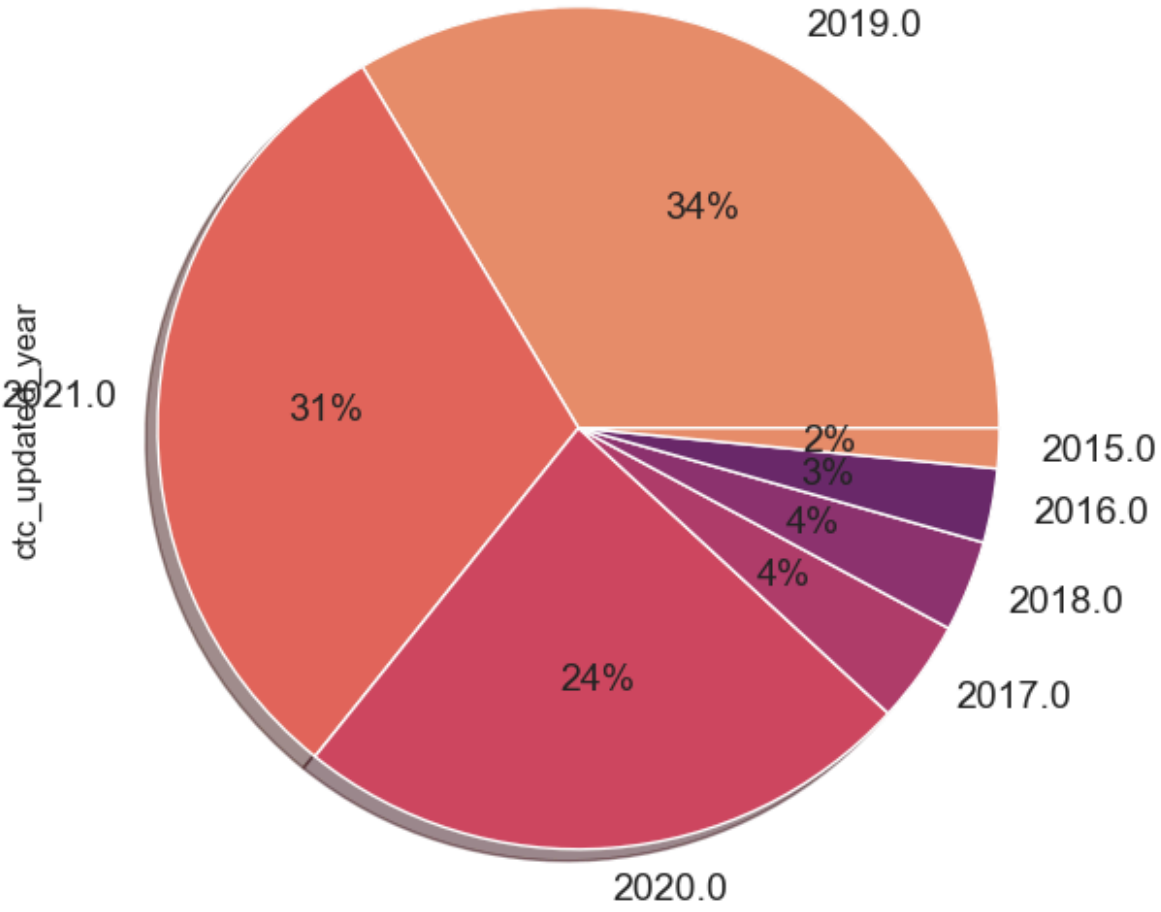
In [497…
```
for i in categorical_cols:
    print(f" Unique values in {i} are {df[i].nunique()}")
```

```
 Unique values in company_hash are 34008
 Unique values in job_position are 762
 Unique values in ctc_updated_year are 7
```

In [ ]:

In [498…
```
# categorical_cols = ['ctc_updated_year']
```

In [499…
```
plt.figure(figsize = (7,8))
count = (df['ctc_updated_year'].value_counts(normalize=True)*100)
count.plot.pie(colors = sns.color_palette("flare"),autopct='%0.0f%%',
                       textprops={'fontsize': 14},shadow = True)
plt.show()
```



In [ ]:

Feature Engineering:

Definition:

1. Designation: Salary an employee is getting wrt salary in the same Company, Job_Position & Years of Experience
2. Class: Salary an employee is getting wrt the salary in the same Company & Job_Position
3. Tier: Salary an employee is getting wrt the salary in the same Company

In [500…

```
# ![image.png](attachment:image.png)

df.company_hash.value_counts()
```

Out[500]:
```
nvnv wgzohrnvzwj otqcxwto    4111
xzegojo                      2910
vbvkgz                       2226
wgszxkvzn                    2115
vwwtznhqt                    1998
                             ...
mvqw xzaxv                      1
wgznghq                         1
uqgbvwn xzegntwy ucn rna        1
bvctqxwpo ftm otqcxwto          1
wyvqntq wgbbhzxwvnxgzo          1
Name: company_hash, Length: 34008, dtype: int64
```

In [ ]:
```
```

In [501…
```
# Masking companies by renaming it to "Others" having count less than 5

df.company_hash.value_counts() <= 5
```

Out[501]:
```
nvnv wgzohrnvzwj otqcxwto    False
xzegojo                      False
vbvkgz                       False
wgszxkvzn                    False
vwwtznhqt                    False
                             ...
mvqw xzaxv                    True
wgznghq                       True
uqgbvwn xzegntwy ucn rna      True
bvctqxwpo ftm otqcxwto        True
wyvqntq wgbbhzxwvnxgzo        True
Name: company_hash, Length: 34008, dtype: bool
```

In [502…
```
df.company_hash.map(df.company_hash.value_counts()) <= 5
```

Out[502]:
```
0         False
1         False
2          True
3         False
4         False
          ...
205836    False
205838    False
205839    False
205840    False
205842    False
Name: company_hash, Length: 168986, dtype: bool
```

In [503…
```
df[df.company_hash.map(df.company_hash.value_counts())<=5]
```

Out[503]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year
2	ojzwnvwnxw vx	2015.0	2000000	backend engineer	2020.0
9	xrbhd	2019.0	360000	NaN	2019.0
11	ngdor ntwy	2016.0	600000	ios engineer	2021.0
16	pnw xzaxv ucn rna	2013.0	800000	other	2020.0
21	axgz srgmvr	2006.0	1550000	engineering leadership	2019.0
...	...	...	...	...	...
205811	mrht onvnt axsxnvr	2013.0	85000	NaN	2016.0
205815	bvptbjnqxu td vbvkgz	2015.0	2400000	NaN	2019.0
205816	wgat ergf ntwy rru	2019.0	2200000	NaN	2020.0
205817	wxowg ojntbo	2011.0	3327000	NaN	2019.0
205834	wyvqntq wgbbhzxwvnxgzo	2020.0	100000	NaN	2019.0

46749 rows × 5 columns

In [ ]:

In [504…
```
df['new']=df.company_hash.mask(df.company_hash.map(df.company_hash.value_counts()))
df['new']
```

Out[504]:
```
0                    atrgxnnt xzaxv
1         qtrxvzwt xzegwgbb rxbxnta
2                               NaN
3                       ngpgutaxv
4                      qxen sqghu
                    ...
205836                    mvqwrvjo
205838                   vuurt xzw
205839                   husqvawgb
205840                    vwwgrxnt
205842                bgqsvz onvzrtj
Name: new, Length: 168986, dtype: object
```

In [505…
```
df
```

Out[505]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	new
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	atrgxnnt xzaxv
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	qtrxvzwt xzegwgbb rxbxnta
2	ojzwnvwnxw vx	2015.0	2000000	backend engineer	2020.0	NaN
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	ngpgutaxv
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	qxen sqghu
...	...	...	...	...	...	...
205836	mvqwrvjo	2011.0	2250000	NaN	2019.0	mvqwrvjo
205838	vuurt xzw	2008.0	220000	NaN	2019.0	vuurt xzw
205839	husqvawgb	2017.0	500000	NaN	2020.0	husqvawgb
205840	vwwgrxnt	2021.0	700000	NaN	2021.0	vwwgrxnt
205842	bgqsvz onvzrtj	2014.0	1240000	NaN	2016.0	bgqsvz onvzrtj

168986 rows × 6 columns

In [ ]:

```
```

In [506…

```
df[df['new']=='Others'].company_hash.value_counts()
```

Out[506]:

```
Series([], Name: company_hash, dtype: int64)
```

In [507…

```
df=df.apply(lambda x: x.mask(x.map(x.value_counts())<=5,'Others') if x.name=='compa
df
```

Out[507]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	new
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	atrgxnnt xzaxv
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	qtrxvzwt xzegwgbb rxbxnta
2	Others	2015.0	2000000	backend engineer	2020.0	NaN
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	ngpgutaxv
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	qxen sqghu
...	...	...	...	...	...	...
205836	mvqwrvjo	2011.0	2250000	NaN	2019.0	mvqwrvjo
205838	vuurt xzw	2008.0	220000	NaN	2019.0	vuurt xzw
205839	husqvawgb	2017.0	500000	NaN	2020.0	husqvawgb
205840	vwwgrxnt	2021.0	700000	NaN	2021.0	vwwgrxnt
205842	bgqsvz onvzrtj	2014.0	1240000	NaN	2016.0	bgqsvz onvzrtj

168986 rows × 6 columns

In [ ]:

In [508…
```
df.company_hash.value_counts()
```

Out[508]:
```
Others                          46749
nvnv wgzohrnvzwj otqcxwto        4111
xzegojo                          2910
vbvkgz                           2226
wgszxkvzn                        2115
                                ...
avowti                              6
ovaart ugxzn ntwyzgrgsxto           6
xzonvzn ojontbo xzw                 6
vrsgowvrt ntwyzgrgsxto xzw          6
ohbngnvr ojontbo                    6
Name: company_hash, Length: 2943, dtype: int64
```

In [509…
```
df.drop(columns='new',inplace=True)
```

In [ ]:

In [510…
```
## Creating Years of Experience Columns
```

In [511…
```
df.drop_duplicates(inplace=True)
df.shape
```

Out[511]:
```
(147140, 5)
```

In [512…
```
#orgyear check
df['orgyear'] = df.apply(lambda x: x['orgyear'] if x['orgyear'] <= 2022 else 2022,
```

In [513…
```
df['years_of_experience']=2022-df['orgyear']
```

In [514…
```
df.drop_duplicates(inplace=True)
df.shape
```

Out[514]:
```
(147101, 6)
```

In [515…
```
df=df[~df['years_of_experience'].isnull()]
```

In [516…
```
#ctc_updated_year_check

df['ctc_updated_year'] = df.apply(lambda x: x['orgyear'] if x['ctc_updated_year'] <
df
```

Out[516]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0
2	Others	2015.0	2000000	backend engineer	2020.0	7.0
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	5.0
…	…	…	…	…	…	…
205836	mvqwrvjo	2011.0	2250000	NaN	2019.0	11.0
205838	vuurt xzw	2008.0	220000	NaN	2019.0	14.0
205839	husqvawgb	2017.0	500000	NaN	2020.0	5.0
205840	vwwgrxnt	2021.0	700000	NaN	2021.0	1.0
205842	bgqsvz onvzrtj	2014.0	1240000	NaN	2016.0	8.0

147101 rows × 6 columns

In [517…
```
#Filling null values with others -- if not done before

df['job_position'] = df['job_position'].fillna('Others')
df['company_hash'] = df['company_hash'].fillna('Others')
```

In [518…
```
missingValue(df)
```

```
Total records in our data =  147101 where missing values are as follows:
```

Out[518]:

	Total Missing	In Percent
company_hash	0	0.0
orgyear	0	0.0
ctc	0	0.0
job_position	0	0.0
ctc_updated_year	0	0.0
years_of_experience	0	0.0

In [519…
```
df.drop_duplicates(inplace=True)
df.shape
```

Out[519]:
```
(146053, 6)
```

In [520…
```
df.describe()
```

Out[520]:

	orgyear	ctc	ctc_updated_year	years_of_experience
count	146053.000000	1.460530e+05	146053.000000	146053.000000
mean	2015.449409	1.129327e+06	2019.598454	6.550591
std	3.300264	7.439632e+05	1.339107	3.300264
min	2006.000000	2.000000e+00	2015.000000	0.000000
25%	2013.000000	5.700000e+05	2019.000000	4.000000
50%	2016.000000	9.600000e+05	2020.000000	6.000000
75%	2018.000000	1.560000e+06	2021.000000	9.000000
max	2022.000000	3.330000e+06	2022.000000	16.000000

In [ ]:
```

```

Manual Clustering based on company, job position and years of experience

In [521…
```
grouped_c_j_y=df.groupby(['years_of_experience','job_position','company_hash'])['ct
```

In [522…
```
grouped_c_j_y
```

Out[522]:

years_of_experience	job_position	company_hash	count	mean	std	min
0.0	Others	Others	42.0	7.058619e+05	674812.642666	200.0
		agzn fgqp xz vzj gqsvzxkvnxgz	1.0	1.600000e+06	NaN	1600000.0
		atrgxnnt	1.0	1.000000e+06	NaN	1000000.0
		atrr	1.0	1.000000e+06	NaN	1000000.0
		atrr ntwyzgrgsxto	2.0	1.000000e+06	282842.712475	800000.0
...	...	...	...	...	...	...
16.0	support engineer	xzegojo	1.0	8.000000e+05	NaN	800000.0
		xzegq	1.0	9.000000e+05	NaN	900000.0
		ywr ntwyzgrgsxto	2.0	8.500000e+05	494974.746831	500000.0
		zvz	1.0	4.000000e+05	NaN	400000.0
	team lead	utqoxontzn ojontbo	1.0	1.600000e+06	NaN	1600000.0

56097 rows × 8 columns

In [523…

```
df_cjy=df.merge(grouped_c_j_y, on=['years_of_experience','job_position','company_ha
df_cjy
```

Out[523]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	co
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0	
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0	
2	Others	2015.0	2000000	backend engineer	2020.0	7.0	45
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0	
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	5.0	
...	...	...	...	...	...	...	
146048	mvqwrvjo	2011.0	2250000	Others	2019.0	11.0	1
146049	vuurt xzw	2008.0	220000	Others	2019.0	14.0	
146050	husqvawgb	2017.0	500000	Others	2020.0	5.0	
146051	vwwgrxnt	2021.0	700000	Others	2021.0	1.0	
146052	bgqsvz onvzrtj	2014.0	1240000	Others	2016.0	8.0	

146053 rows × 14 columns

In [524…

```
df_cjy.sort_values(['years_of_experience','job_position','company_hash'])
```

Out[524]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	cou
896	Others	2022.0	120000	Others	2022.0	0.0	42
2599	Others	2022.0	430000	Others	2022.0	0.0	42
7691	Others	2022.0	570000	Others	2022.0	0.0	42
7870	Others	2022.0	550000	Others	2022.0	0.0	42
8789	Others	2022.0	680000	Others	2022.0	0.0	42
...	...	...	...	...	...	...	
73608	xzegq	2006.0	900000	support engineer	2021.0	16.0	1
11355	ywr ntwyzgrgsxto	2006.0	500000	support engineer	2021.0	16.0	2
37161	ywr ntwyzgrgsxto	2006.0	1200000	support engineer	2021.0	16.0	2
14265	zvz	2006.0	400000	support engineer	2021.0	16.0	1
59644	utqoxontzn ojontbo	2006.0	1600000	team lead	2021.0	16.0	1

146053 rows × 14 columns

In [525…
```
#no change till now

df_cjy.drop_duplicates(inplace=True)
df_cjy.shape
```

Out[525]:
```
(146053, 14)
```

In [526…
```
## Creating Designation basis on the salary they are getting in their respective cc

def condition_designation(a,b_50,b_75):
    if a<b_50:
        return 3
    elif a>=b_50 and a<=b_75:
        return 2
    elif a>=b_75:
        return 1
```

In [527…
```
df.head()
```

Out[527]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0
2	Others	2015.0	2000000	backend engineer	2020.0	7.0
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	5.0

In [528…
```
df_cjy['designation'] =df_cjy.apply(lambda x: condition_designation(x['ctc'],x['50%
```

In [529…
```
df_cjy.head()
```

Out[529]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	count	
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0	1.0	1
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0	7.0	7
2	Others	2015.0	2000000	backend engineer	2020.0	7.0	456.0	9
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0	7.0	1
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	5.0	1.0	1

In [530…
```
df_cjy.shape
```

Out[530]:
```
(146053, 15)
```

In [531…
```
df_cjy.designation.value_counts(normalize=True)*100
```

Out[531]:
```
2    44.119600
3    34.180058
1    21.700342
Name: designation, dtype: float64
```

In [ ]:
```
```

Manual Clustering based on company and job position

In [532…
```
grouped_c_j=df.groupby(['job_position','company_hash'])['ctc'].describe()
```

In [533…
```
grouped_c_j
```

Out[533]:

job_position	company_hash	count	mean	std	min	25%	50%	
Others	**Others**	3159.0	1.025099e+06	837191.520717	15.0	358500.0	800000.0	1
	a ntwyzgrgsxto	5.0	6.750000e+05	389711.431703	350000.0	500000.0	575000.0	
	aaqxctz avnv owxtzwto vzvrjnxwo ucn rna	1.0	5.000000e+05	NaN	500000.0	500000.0	500000.0	
	adw ntwyzgrgsj	59.0	6.451864e+05	449039.606370	80000.0	374000.0	500000.0	
	adw ntwyzgrgsxto	37.0	6.230000e+05	323412.705035	100000.0	400000.0	525000.0	
...	**...**	...	...	...	...	...	...	
wordpress developer	**Others**	1.0	6.000000e+05	NaN	600000.0	600000.0	600000.0	
worker	**zgn vuurxwvmrt vwwghzn**	1.0	2.000000e+05	NaN	200000.0	200000.0	200000.0	
x	**Others**	1.0	4.000000e+05	NaN	400000.0	400000.0	400000.0	
young professional ii	**sgctqzbtzn ge xzaxv**	1.0	5.000000e+05	NaN	500000.0	500000.0	500000.0	
zomato	**kgbvng**	1.0	5.000000e+05	NaN	500000.0	500000.0	500000.0	

21596 rows × 8 columns

In [534…
```
df.drop_duplicates().shape
```

Out[534]:
```
(146053, 6)
```

In [535…
```
df_cj=df.merge(grouped_c_j, on=['job_position','company_hash'], how='left')
df_cj
```

Out[535]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	co
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0	
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0	
2	Others	2015.0	2000000	backend engineer	2020.0	7.0	38
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0	
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	5.0	
...	...	...	...	...	...	...	
146048	mvqwrvjo	2011.0	2250000	Others	2019.0	11.0	
146049	vuurt xzw	2008.0	220000	Others	2019.0	14.0	
146050	husqvawgb	2017.0	500000	Others	2020.0	5.0	
146051	vwwgrxnt	2021.0	700000	Others	2021.0	1.0	
146052	bgqsvz onvzrtj	2014.0	1240000	Others	2016.0	8.0	1

146053 rows × 14 columns

In [536…
```
df_cj.sort_values(['company_hash','job_position','years_of_experience'])
```

Out[536]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	co
896	Others	2022.0	120000	Others	2022.0	0.0	31
2599	Others	2022.0	430000	Others	2022.0	0.0	31
7691	Others	2022.0	570000	Others	2022.0	0.0	31
7870	Others	2022.0	550000	Others	2022.0	0.0	31
8789	Others	2022.0	680000	Others	2022.0	0.0	31
...	...	...	...	...	...	...	
122135	zxztrtvuo	2013.0	1200000	ios engineer	2017.0	9.0	
53733	zxztrtvuo	2016.0	1200000	member of technical staff at nineleaps	2020.0	6.0	
9189	zxztrtvuo	2020.0	450000	other	2020.0	2.0	
133204	zxztrtvuo	2019.0	450000	other	2020.0	3.0	
37242	zxztrtvuo	2016.0	1200000	software developer intern	2020.0	6.0	

146053 rows × 14 columns

In [537…
```
df_cj.shape
```

Out[537]: (146053, 14)

In [538…
```
df_cj.drop_duplicates(inplace=True)
```

In [539…
```
df_cj.shape
```

Out[539]: (146053, 14)

Creating Class basis on the salary they are getting in their respective company

In [540…
```
def condition_classs(a,b_50,b_75):
    if a<b_50:
        return 3
    elif a>=b_50 and a<=b_75:
        return 2

    elif a>=b_75:
        return 1
```

In [541…
```
df_cj['classs'] =df_cj.apply(lambda x: condition_classs(x['ctc'],x['50%'],x['75%'])
df_cj
```

Out[541]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	co
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0	
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0	2
2	Others	2015.0	2000000	backend engineer	2020.0	7.0	38
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0	2
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	5.0	
...	...	...	...	...	...	...	
146048	mvqwrvjo	2011.0	2250000	Others	2019.0	11.0	6
146049	vuurt xzw	2008.0	220000	Others	2019.0	14.0	1
146050	husqvawgb	2017.0	500000	Others	2020.0	5.0	1
146051	vwwgrxnt	2021.0	700000	Others	2021.0	1.0	3
146052	bgqsvz onvzrtj	2014.0	1240000	Others	2016.0	8.0	10

146053 rows × 15 columns

In [542…
```
df_cj.classs.value_counts(normalize=True)*100
```

Out[542]:
```
3    43.736178
2    31.830911
1    24.432911
Name: classs, dtype: float64
```

In [543…
```
# job position that has the highest class

df_cj[df_cj['classs']==1][['job_position','ctc']].groupby('job_position')['ctc'].de
```

Out[543]:

job_position	count	mean	std	min	25%	50%	75%	
Others	8217.0	1.931143e+06	695531.136886	100000.0	1400000.0	1900000.0	2500000.0	33
android engineer	913.0	1.784897e+06	638704.770985	14000.0	1320000.0	1700000.0	2200000.0	33
application developer	1.0	1.150000e+06	NaN	1150000.0	1150000.0	1150000.0	1150000.0	11
application developer analyst	1.0	6.000000e+05	NaN	600000.0	600000.0	600000.0	600000.0	6
application development analyst	2.0	8.150000e+05	233345.237792	650000.0	732500.0	815000.0	897500.0	9
...	...	...	...	...	...	...	...	
support engineer	683.0	1.190779e+06	552019.578789	350000.0	830000.0	1000000.0	1400000.0	33
system engineer	10.0	8.420000e+05	373118.986086	400000.0	550000.0	775000.0	1100000.0	15
teaching assistant	1.0	1.800000e+06	NaN	1800000.0	1800000.0	1800000.0	1800000.0	18
team lead	2.0	1.800000e+06	565685.424949	1400000.0	1600000.0	1800000.0	2000000.0	22
technology analyst	3.0	8.966667e+05	351046.055858	660000.0	695000.0	730000.0	1015000.0	13

108 rows × 8 columns

In [544…
```
df_cjy.head()
```

Out[544]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	count	
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0	1.0	1
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0	7.0	7
2	Others	2015.0	2000000	backend engineer	2020.0	7.0	456.0	9
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0	7.0	1
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	5.0	1.0	1

In [545…
```
df_cj.head()
```

Out[545]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	count	
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0	2.0	1
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0	25.0	9
2	Others	2015.0	2000000	backend engineer	2020.0	7.0	3871.0	1
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0	24.0	1
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	5.0	3.0	8

In [546…
```
df_cj.shape
```

Out[546]:
```
(146053, 15)
```

In [547…
```
df_cjy.shape
```

Out[547]:
```
(146053, 15)
```

In [548…
```
df_cj.drop(columns=['count','mean','std','min','25%','50%','75%','max'],inplace=Tru
df_cjy.drop(columns=['count','mean','std','min','25%','50%','75%','max'],inplace=Tr
```

In [549…
```
df_cj.drop_duplicates().shape
```

Out[549]:
```
(146053, 7)
```

In [550…
```
df_cjy.drop_duplicates().shape
```

Out[550]:
```
(146053, 7)
```

In [551…
```
df_cjy
```

Out[551]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	de
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0	
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0	
2	Others	2015.0	2000000	backend engineer	2020.0	7.0	
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0	
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	5.0	
...	...	...	...	...	...	...	
146048	mvqwrvjo	2011.0	2250000	Others	2019.0	11.0	
146049	vuurt xzw	2008.0	220000	Others	2019.0	14.0	
146050	husqvawgb	2017.0	500000	Others	2020.0	5.0	
146051	vwwgrxnt	2021.0	700000	Others	2021.0	1.0	
146052	bgqsvz onvzrtj	2014.0	1240000	Others	2016.0	8.0	

146053 rows × 7 columns

In [552…

```
df_cj
```

Out[552]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	cla
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0	
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0	
2	Others	2015.0	2000000	backend engineer	2020.0	7.0	
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0	
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	5.0	
...	...	...	...	...	...	...	
146048	mvqwrvjo	2011.0	2250000	Others	2019.0	11.0	
146049	vuurt xzw	2008.0	220000	Others	2019.0	14.0	
146050	husqvawgb	2017.0	500000	Others	2020.0	5.0	
146051	vwwgrxnt	2021.0	700000	Others	2021.0	1.0	
146052	bgqsvz onvzrtj	2014.0	1240000	Others	2016.0	8.0	

146053 rows × 7 columns

In [553…
```
df_cjy_cj=df_cj.merge(df_cjy, on=['company_hash','orgyear','ctc','job_position','ye
df_cjy_cj
```

Out[553]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	cla
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0	
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0	
2	Others	2015.0	2000000	backend engineer	2020.0	7.0	
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0	
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	5.0	
...	...	...	...	...	...	...	
146048	mvqwrvjo	2011.0	2250000	Others	2019.0	11.0	
146049	vuurt xzw	2008.0	220000	Others	2019.0	14.0	
146050	husqvawgb	2017.0	500000	Others	2020.0	5.0	
146051	vwwgrxnt	2021.0	700000	Others	2021.0	1.0	
146052	bgqsvz onvzrtj	2014.0	1240000	Others	2016.0	8.0	

146053 rows × 8 columns

In [554…
```
df_cjy_cj.shape
```

Out[554]:
```
(146053, 8)
```

In [555…
```
df_cjy_cj.drop_duplicates().shape
```

Out[555]:
```
(146053, 8)
```

In [ ]:
```

```

Manual Clustering based on company

In [556…
```
grouped_c=df.groupby(['company_hash'])['ctc'].describe()
```

In [557…
```
df_c=df.merge(grouped_c, on=['company_hash'], how='left')
```

In [558…
```
df_c.head(5)
```

Out[558]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	count
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0	9.0
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0	384.0
2	Others	2015.0	2000000	backend engineer	2020.0	7.0	24489.0
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0	59.0
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	5.0	6.0

In [559…

```
#verify

df_c.sort_values(['company_hash'])
```

Out[559]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	c
87851	Others	2007.0	1300000	backend engineer	2019.0	15.0	24
70542	Others	2016.0	520000	engineering intern	2016.0	6.0	24
70547	Others	2016.0	1140000	backend engineer	2020.0	6.0	24
70548	Others	2018.0	360000	Others	2019.0	4.0	24
70561	Others	2019.0	2000000	research engineers	2019.0	3.0	24
...	...	...	...	...	...	...	
130708	zxztrtvuo	2016.0	720000	backend engineer	2019.0	6.0	
122135	zxztrtvuo	2013.0	1200000	ios engineer	2017.0	9.0	
60666	zxztrtvuo	2018.0	1440000	Others	2020.0	4.0	
119075	zxztrtvuo	2017.0	900000	backend engineer	2020.0	5.0	
118793	zxztrtvuo	2016.0	930000	frontend engineer	2019.0	6.0	

146053 rows × 14 columns

In [560…

```
print(df.drop_duplicates().shape)
print(df_c.shape)
print(df_c.drop_duplicates().shape)
```

```
(146053, 6)
(146053, 14)
(146053, 14)
```

In [ ]:

Creating Tier basis on the salary in the companies

In [561…
```
def condition_tier(a,b_50,b_75):
    if a<b_50:
        return 3
    elif a>=b_50 and a<=b_75:
        return 2
    elif a>=b_75:
        return 1
```

In [562…
```
df_c['tier'] =df_c.apply(lambda x: condition_tier(x['ctc'],x['50%'],x['75%']),axis
df_c
```

Out[562]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	c
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0	
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0	
2	Others	2015.0	2000000	backend engineer	2020.0	7.0	24
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0	
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	5.0	
...	...	...	...	...	...	...	
146048	mvqwrvjo	2011.0	2250000	Others	2019.0	11.0	
146049	vuurt xzw	2008.0	220000	Others	2019.0	14.0	
146050	husqvawgb	2017.0	500000	Others	2020.0	5.0	
146051	vwwgrxnt	2021.0	700000	Others	2021.0	1.0	
146052	bgqsvz onvzrtj	2014.0	1240000	Others	2016.0	8.0	

146053 rows × 15 columns

In [563…
```
df_c.head()
```

Out[563]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	count
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0	9.0
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0	384.0
2	Others	2015.0	2000000	backend engineer	2020.0	7.0	24489.0
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0	59.0
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	5.0	6.0

In [564…

```
df_c.tier.value_counts(normalize=True)*100
```

Out[564]:

```
3    47.952456
2    28.153479
1    23.894066
Name: tier, dtype: float64
```

In [565…

```
df_cjy_cj_c=df_cjy_cj.merge(df_c, on=['company_hash','orgyear','ctc','job_position'
                                      ,'ctc_updated_year','years_of_experience'], h
```

In [566…

```
df_cjy_cj_c.head(10)
```

Out[566]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	classs	d
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0	1	
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0	3	
2	Others	2015.0	2000000	backend engineer	2020.0	7.0	1	
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0	3	
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	5.0	1	
5	yvuuxrj hzbvqqxta bvqptnxzs ucn rna	2018.0	700000	fullstack engineer	2020.0	4.0	2	
6	lubgqsvz wyvot wg	2018.0	1500000	fullstack engineer	2019.0	4.0	3	
7	vwwtznhqt ntwyzgrgsj	2019.0	400000	backend engineer	2019.0	3.0	3	
8	utqoxontzn ojontbo	2020.0	450000	Others	2020.0	2.0	3	
9	Others	2019.0	360000	Others	2019.0	3.0	3	

In [567…

```
df_cjy_cj_c.shape
```

Out[567]: (146053, 17)

In [568…
```
data=df_cjy_cj_c.copy(deep=True)
```

In [569…
```
data.drop(columns=['count','mean','std','min','25%','50%','75%','max'],inplace=True
```

In [570…
```
data
```

Out[570]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	cla
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0	
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0	
2	Others	2015.0	2000000	backend engineer	2020.0	7.0	
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0	
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	5.0	
...	...	...	...	...	...	...	
146048	mvqwrvjo	2011.0	2250000	Others	2019.0	11.0	
146049	vuurt xzw	2008.0	220000	Others	2019.0	14.0	
146050	husqvawgb	2017.0	500000	Others	2020.0	5.0	
146051	vwwgrxnt	2021.0	700000	Others	2021.0	1.0	
146052	bgqsvz onvzrtj	2014.0	1240000	Others	2016.0	8.0	

146053 rows × 9 columns

In [571…
```
# org_data = pd.read_csv('data/scaler_clustering.csv')
# org_data
```

In [572…
```
# df_new=data.merge(org_data, on=['company_hash'], how = 'left')
# df_new
```

In [573…
```
pd.set_option('display.max_rows', 20)
```

In [574…
```
# Top 10 companies providing highest ctc's

data.groupby(['company_hash'])['ctc'].max().head(11).sort_values(ascending = False)
```

Out[574]:
```
company_hash
Others                                          3329999
adw ntwyzgrgsj                                  3200000
a ntwyzgrgsxto                                  3150000
agnut                                           3000000
agdutq                                          2500000
aghmnzhn                                        2400000
adw ntwyzgrgsxto                                2350000
agotrtwn                                        1610000
agnoihvqto                                      1600000
aaqxctz avnv owxtzwto  vzvrjnxwo ucn rna        1400000
aggartmrht xzzgcvnxgzo                          1000000
Name: ctc, dtype: int64
```

In [ ]:

Overview of what's next :

1. Data processing for Unsupervised clustering - Label encoding/ One- hot encoding, Standardization of data
2. Unsupervised Learning - Clustering
3. Checking clustering tendency
4. Elbow method
5. K-means clustering
6. Hierarchical clustering
7. Insights from Unsupervised Clustering
8. Provide actionable Insights & Recommendations for the Business.

1. K-Means is a distance-based algorithm. Because of that, it's really important to perform feature scaling (normalize, standardize, or choose any other option in which the distance has some comparable meaning for all the columns).
2. In this example, we use MinMaxScaler instead of StandardScaler, so as to transforming the feature values to fall within the bounded intervals (min and max), rather than making them to fall around mean as 0 with standard deviation as 1 (StandardScaler).
3. MinMaxScaler is an excellent tool for this purpose. MinMaxScaler scales all the data features in the range [0, 1] or else in the range [-1, 1] if there are negative values in the dataset. This scaling compresses all the inliers in the narrow range [0, 0.005].

In [ ]:

In [575…
```
data.shape
```

Out[575]:
```
(146053, 9)
```

In [576…
```
data['company_hash'].unique()
```

Out[576]:
```
array(['atrgxnnt xzaxv', 'qtrxvzwt xzegwgbb rxbxnta', 'Others', ...,
       'srgxej', 'bh oxsbv', 'ohbngnvr ojontbo'], dtype=object)
```

In [577…
```
# Label Encoding
```

```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
data['company_hash']= label_encoder.fit_transform(data['company_hash'])
data['company_hash'].unique()
```

Out[577]:
```
array([  45, 1497,    0, ..., 1667,  138, 1155])
```

In [578…
```
data['job_position']= label_encoder.fit_transform(data['job_position'])
len(data['job_position'].unique())
```

Out[578]:
```
763
```

In [579…
```
data
```

Out[579]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	cla
0	45	2016.0	1100000	378	2020.0	6.0	
1	1497	2018.0	449999	235	2019.0	4.0	
2	0	2015.0	2000000	105	2020.0	7.0	
3	936	2017.0	700000	105	2019.0	5.0	
4	1535	2017.0	1400000	235	2019.0	5.0	
...	...	...	...	...	...	...	
146048	884	2011.0	2250000	0	2019.0	11.0	
146049	2158	2008.0	220000	0	2019.0	14.0	
146050	636	2017.0	500000	0	2020.0	5.0	
146051	2186	2021.0	700000	0	2021.0	1.0	
146052	127	2014.0	1240000	0	2016.0	8.0	

146053 rows × 9 columns

In [580…
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 146053 entries, 0 to 146052
Data columns (total 9 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   company_hash         146053 non-null  int32
 1   orgyear              146053 non-null  float64
 2   ctc                  146053 non-null  int64
 3   job_position         146053 non-null  int32
 4   ctc_updated_year     146053 non-null  float64
 5   years_of_experience  146053 non-null  float64
 6   classs               146053 non-null  int64
 7   designation          146053 non-null  int64
 8   tier                 146053 non-null  int64
dtypes: float64(3), int32(2), int64(4)
memory usage: 10.0 MB
```

In [581…
```
# dropping org year and cts_updated year as we already have years of experience

data.drop(columns=['orgyear'],inplace=True)
data.drop(columns=['ctc_updated_year'],inplace=True)
```

In [582…
```
missingValue(data)
```

```
Total records in our data =  146053 where missing values are as follows:
```

Out[582]:

	Total Missing	In Percent
company_hash	0	0.0
ctc	0	0.0
job_position	0	0.0
years_of_experience	0	0.0
classs	0	0.0
designation	0	0.0
tier	0	0.0

In [583…
```
data.head()
```

Out[583]:

	company_hash	ctc	job_position	years_of_experience	classs	designation	tier
0	45	1100000	378	6.0	1	2	2
1	1497	449999	235	4.0	3	3	3
2	0	2000000	105	7.0	1	1	1
3	936	700000	105	5.0	3	3	3
4	1535	1400000	235	5.0	1	2	1

In [584…
```
# Creating second copy after org_df

data_1 = data.copy()
```

In [585…
```
from sklearn.preprocessing import MinMaxScaler

# scaler = MinMaxScaler()
# scaler.fit(data)
# data=scaler.transform(data)

ms = MinMaxScaler()

data[['ctc']] = ms.fit_transform(data[['ctc']])
data.head()
```

Out[585]:

	company_hash	ctc	job_position	years_of_experience	classs	designation	tier
0	45	0.330330	378	6.0	1	2	2
1	1497	0.135134	235	4.0	3	3	3
2	0	0.600600	105	7.0	1	1	1
3	936	0.210210	105	5.0	3	3	3
4	1535	0.420420	235	5.0	1	2	1

In [ ]:
```
```

Clustering using Sklearn's implementation of Kmeans

In [586…
```
# from sklearn.cluster import KMeans

# k = 3 ## arbitrary value
# kmeans = KMeans(n_clusters=k)
# y_pred = kmeans.fit_predict(data)
```

In [587…
```
# ## what are learned labels(cluster #)
# y_pred
```

In [588…
```
# ##coordinates of the cluster centers
# kmeans.cluster_centers_
```

In [589…
```
X = data_1.copy()
scaler = MinMaxScaler()
scaler.fit(X)
X=scaler.transform(X)
```

In [590…
```
from sklearn.cluster import KMeans

k = 3 ## arbitrary value
kmeans = KMeans(n_clusters=k)
y_pred = kmeans.fit_predict(X)
```

In [591…
```
##coordinates of the cluster centers

kmeans.cluster_centers_
```

Out[591]:
```
array([[0.42121771, 0.58211838, 0.22688338, 0.49832136, 0.0794803 ,
        0.22844558, 0.05967212],
       [0.41462362, 0.18998503, 0.23260469, 0.35999868, 0.98794806,
        0.86552791, 0.9793948 ],
       [0.45037175, 0.32969372, 0.25721031, 0.399593  , 0.52533483,
        0.45442012, 0.62362649]])
```

In [592…
```
y_pred is kmeans.labels_
```

Out[592]:
```
True
```

In [ ]:
```

```

Visualizing Sklearn Clusters

In [593…
```
X
```

Out[593]:
```
array([[0.01529572, 0.33032993, 0.49606299, ..., 0.        , 0.5       ,
        0.5       ],
       [0.50883753, 0.13513432, 0.30839895, ..., 1.        , 1.        ,
        1.        ],
       [0.        , 0.60060036, 0.13779528, ..., 0.        , 0.        ,
        0.        ],
       ...,
       [0.21617947, 0.15014964, 0.        , ..., 1.        , 1.        ,
        1.        ],
       [0.74303195, 0.21020974, 0.        , ..., 1.        , 0.5       ,
        1.        ],
       [0.04316791, 0.372372  , 0.        , ..., 1.        , 1.        ,
        1.        ]])
```

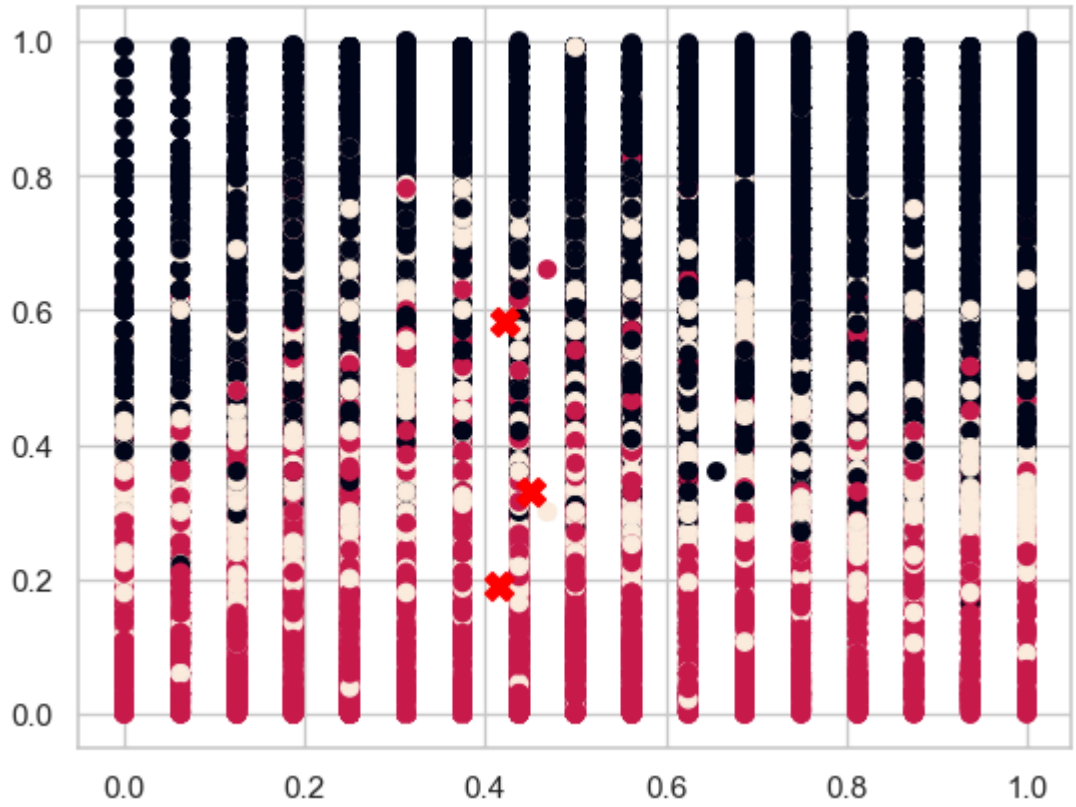
In [594…
```
clusters = pd.DataFrame(X, columns=data_1.columns)
clusters['label'] = kmeans.labels_
clusters
```

Out[594]:

	company_hash	ctc	job_position	years_of_experience	classs	designation	tier	labe
0	0.015296	0.330330	0.496063	0.3750	0.0	0.5	0.5	2
1	0.508838	0.135134	0.308399	0.2500	1.0	1.0	1.0	1
2	0.000000	0.600600	0.137795	0.4375	0.0	0.0	0.0	0
3	0.318151	0.210210	0.137795	0.3125	1.0	1.0	1.0	1
4	0.521754	0.420420	0.308399	0.3125	0.0	0.5	0.0	0
...	...	...	...	...	...	...	...	...
146048	0.300476	0.675675	0.000000	0.6875	0.0	0.0	0.0	0
146049	0.733515	0.066066	0.000000	0.8750	1.0	0.5	1.0	1
146050	0.216179	0.150150	0.000000	0.3125	1.0	1.0	1.0	1
146051	0.743032	0.210210	0.000000	0.0625	1.0	0.5	1.0	1
146052	0.043168	0.372372	0.000000	0.5000	1.0	1.0	1.0	1

146053 rows × 8 columns

In [595…
```
def viz_clusters(kmeans):
    plt.scatter(clusters['years_of_experience'], clusters['ctc'], c=clusters['label
    plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
                color="red",
                marker="X",
                s=100)

viz_clusters(kmeans)
```



Comments:

1. I have tried bunch of features to visualize but I am not able to get proper result and it's not clear.
2. Using polar plot for better visialization.

In [596…

```
# Using polar plot for better visialization:

polar = clusters.groupby("label").mean().reset_index()
polar = pd.melt(polar, id_vars=["label"])
polar
```

Out[596]:

	label	variable	value
0	0	company_hash	0.421182
1	1	company_hash	0.414624
2	2	company_hash	0.450414
3	0	ctc	0.582032
4	1	ctc	0.189985
...	...	...	...
16	1	designation	0.865528
17	2	designation	0.454336
18	0	tier	0.059832
19	1	tier	0.979395
20	2	tier	0.623768

21 rows × 3 columns

In [597…
```
# pip install plotly
```

In [ ]:
```
import plotly.express as px

fig = px.line_polar(polar, r="value", theta="variable", color="label", line_close=T
fig.show()
```

Feature definitions:

1. Designation: Salary an employee is getting wrt salary in the same Company, Job_Position & Years of Experience
2. Class: Salary an employee is getting wrt the salary in the same Company & Job_Position
3. Tier: Salary an employee is getting wrt the salary in the same Companydescent

Observations:

1. We have three cluster mainly (label - 0, 1, 2) job_position , years of experience, comapny_hash for all the people in the
   three cluster is nearly same. So we can compare the other features keeping this useful info in mind.
2. The students whose salaries are already high (Label 2), and who comes from a descent job role in a descent company, having slightly more amount experience, hardly care about designation , class or tier as they all are best of all !!
3. The students who have median salary (not too high, not too low) (Label 0), and who comes from a descent job role in a
   descent company, having descent amount experience, requires little upscalling.
4. The students who have least salary (Label 1), and who comes from a descent job role in a descent company, having descent
   amount experience, requires lots of upscalling. As these students belongs to designation - 3, class- 3, tier- 3

Recommendations:

1. Scaler should completely ignore 'Label 2' students for advertising/marketing their product as they don't need to upskill as they already are super skilled.
2. Instead, Scaler team should identify and talk to these folks if they are interested in teaching/mentoring. This way, Scaler would be having best of the best instructors/mentors in the business.
3. Scaler should advertise to 'Label 0' set of students with some advanced courses so that they can compete with top tier students.
4. 'Label 1' are the target audience. Scaler team should heavily focus on advertising / marketing all their tech products/
   couses, free master clases, to these set of learners

In [ ]:
```

```

In [602…
```
data_new = data.copy()
```

In [603…
```
data_new.dropna(inplace=True)
```

In [604…
```
data_new
```

Out[604]:

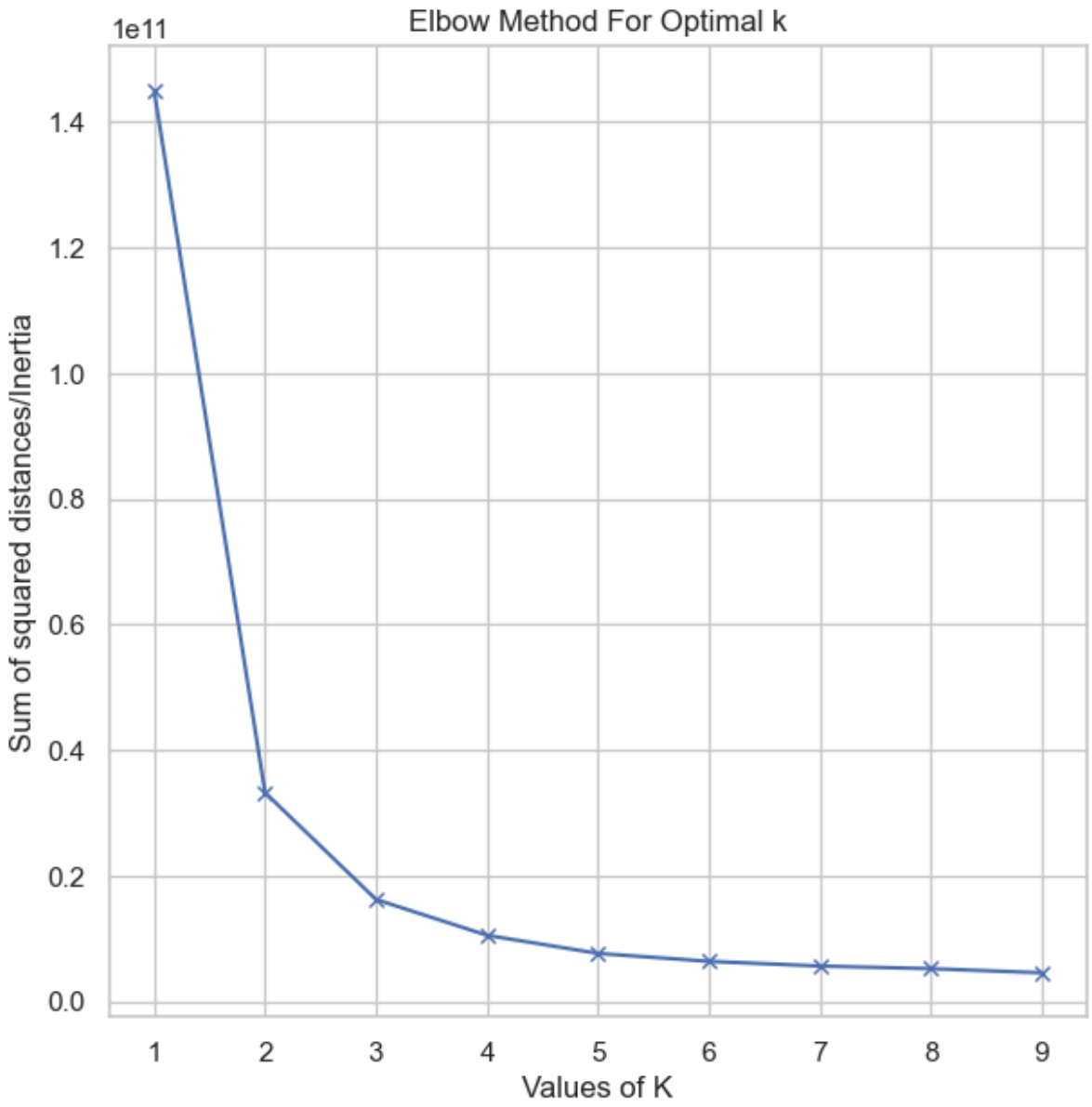
	company_hash	ctc	job_position	years_of_experience	classs	designation	tier
0	45	0.330330	378	6.0	1	2	2
1	1497	0.135134	235	4.0	3	3	3
2	0	0.600600	105	7.0	1	1	1
3	936	0.210210	105	5.0	3	3	3
4	1535	0.420420	235	5.0	1	2	1
...	...	...	...	...	...	...	...
146048	884	0.675675	0	11.0	1	1	1
146049	2158	0.066066	0	14.0	3	2	3
146050	636	0.150150	0	5.0	3	3	3
146051	2186	0.210210	0	1.0	3	2	3
146052	127	0.372372	0	8.0	3	3	3

146053 rows × 7 columns

In [ ]:
```
```

Elbow Method

In [605…
```
plt.figure(figsize = (7,7))
Sum_of_squared_distances = []
K = range(1,10)
for num_clusters in K :
    kmeans = KMeans(n_clusters=num_clusters)
    kmeans.fit(data_new)
    Sum_of_squared_distances.append(kmeans.inertia_)
plt.plot(K,Sum_of_squared_distances,'bx-')
plt.xlabel('Values of K')
plt.ylabel('Sum of squared distances/Inertia')
plt.title('Elbow Method For Optimal k')
plt.show()
```



Comments:

From above plot, it is clear that we require 3 clusters and our earlier assumption is correct.

In [ ]:

In [606…
```
kmeans = KMeans(n_clusters=3)
kmeans.fit(data_new)
print(kmeans.cluster_centers_)
print(kmeans.cluster_centers_.shape)
```

```
[[2.41600065e+03 3.28781567e-01 1.79098928e+02 6.30844693e+00
  2.19058782e+00 2.11382082e+00 2.24198251e+00]
 [2.20004364e+02 3.38710849e-01 1.89660587e+02 6.82628404e+00
  2.20934977e+00 2.16183802e+00 2.24353766e+00]
 [1.25705244e+03 3.53259178e-01 1.75784246e+02 6.46716679e+00
  2.17259274e+00 2.08547426e+00 2.23448258e+00]]
(3, 7)
```

In [607…
```
data_new['k-m label']=kmeans.fit_predict(data_new)
```

In [608…
```
data_new
```

Out[608]:

	company_hash	ctc	job_position	years_of_experience	classs	designation	tier	k-m labe
0	45	0.330330	378	6.0	1	2	2	
1	1497	0.135134	235	4.0	3	3	3	
2	0	0.600600	105	7.0	1	1	1	
3	936	0.210210	105	5.0	3	3	3	
4	1535	0.420420	235	5.0	1	2	1	
...	...	...	...	...	...	...	...	
146048	884	0.675675	0	11.0	1	1	1	
146049	2158	0.066066	0	14.0	3	2	3	
146050	636	0.150150	0	5.0	3	3	3	
146051	2186	0.210210	0	1.0	3	2	3	
146052	127	0.372372	0	8.0	3	3	3	

146053 rows × 8 columns

In [609…

```
data_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 146053 entries, 0 to 146052
Data columns (total 8 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   company_hash         146053 non-null  int32  
 1   ctc                  146053 non-null  float64
 2   job_position         146053 non-null  int32  
 3   years_of_experience  146053 non-null  float64
 4   classs               146053 non-null  int64  
 5   designation          146053 non-null  int64  
 6   tier                 146053 non-null  int64  
 7   k-m label            146053 non-null  int32  
dtypes: float64(2), int32(3), int64(3)
memory usage: 8.4 MB
```

In [610…

```
df_cjy_cj_c
```

Out[610]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	cla
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0	
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0	
2	Others	2015.0	2000000	backend engineer	2020.0	7.0	
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0	
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	5.0	
...	...	...	...	...	...	...	
146048	mvqwrvjo	2011.0	2250000	Others	2019.0	11.0	
146049	vuurt xzw	2008.0	220000	Others	2019.0	14.0	
146050	husqvawgb	2017.0	500000	Others	2020.0	5.0	
146051	vwwgrxnt	2021.0	700000	Others	2021.0	1.0	
146052	bgqsvz onvzrtj	2014.0	1240000	Others	2016.0	8.0	

146053 rows × 17 columns

In [611…
```
df_cjy_cj_c.drop(columns=['count','mean','std','min','25%','50%','75%','max'],inpla
```

In [612…
```
data_org = df_cjy_cj_c.copy()
```

In [613…
```
final_data = pd.concat([data_org,data_new['k-m label']], axis=1)
final_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 146053 entries, 0 to 146052
Data columns (total 10 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   company_hash         146053 non-null  object 
 1   orgyear              146053 non-null  float64
 2   ctc                  146053 non-null  int64  
 3   job_position         146053 non-null  object 
 4   ctc_updated_year     146053 non-null  float64
 5   years_of_experience  146053 non-null  float64
 6   classs               146053 non-null  int64  
 7   designation          146053 non-null  int64  
 8   tier                 146053 non-null  int64  
 9   k-m label            146053 non-null  int32  
dtypes: float64(3), int32(1), int64(4), object(2)
memory usage: 11.7+ MB
```

In [614…
```
final_data
```

Out[614]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	cla
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0	
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0	
2	Others	2015.0	2000000	backend engineer	2020.0	7.0	
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0	
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	5.0	
...	...	...	...	...	...	...	
146048	mvqwrvjo	2011.0	2250000	Others	2019.0	11.0	
146049	vuurt xzw	2008.0	220000	Others	2019.0	14.0	
146050	husqvawgb	2017.0	500000	Others	2020.0	5.0	
146051	vwwgrxnt	2021.0	700000	Others	2021.0	1.0	
146052	bgqsvz onvzrtj	2014.0	1240000	Others	2016.0	8.0	

146053 rows × 10 columns

In [ ]:

In [615…
```
#the most we could do without crashing

data_frac=data_new.sample(frac=0.0025)
```

In [616…
```
data_frac
```

Out[616]:

	company_hash	ctc	job_position	years_of_experience	classs	designation	tier	k-m labe
104455	2835	0.180180	378	4.0	2	2	3	1
13096	1058	0.210210	478	2.0	2	2	2	2
75361	186	0.405405	478	10.0	1	2	1	0
86532	1200	0.600600	0	6.0	1	1	1	2
40848	1716	0.151351	162	6.0	3	2	3	2
...	...	...	...	...	...	...	...	...
96030	752	0.069069	105	7.0	3	1	3	2
9216	1509	0.150150	0	4.0	3	3	3	2
142311	644	0.330330	378	8.0	1	2	3	0
91327	0	0.219219	230	6.0	3	2	3	0
80797	0	0.156156	378	0.0	3	3	3	0

365 rows × 8 columns

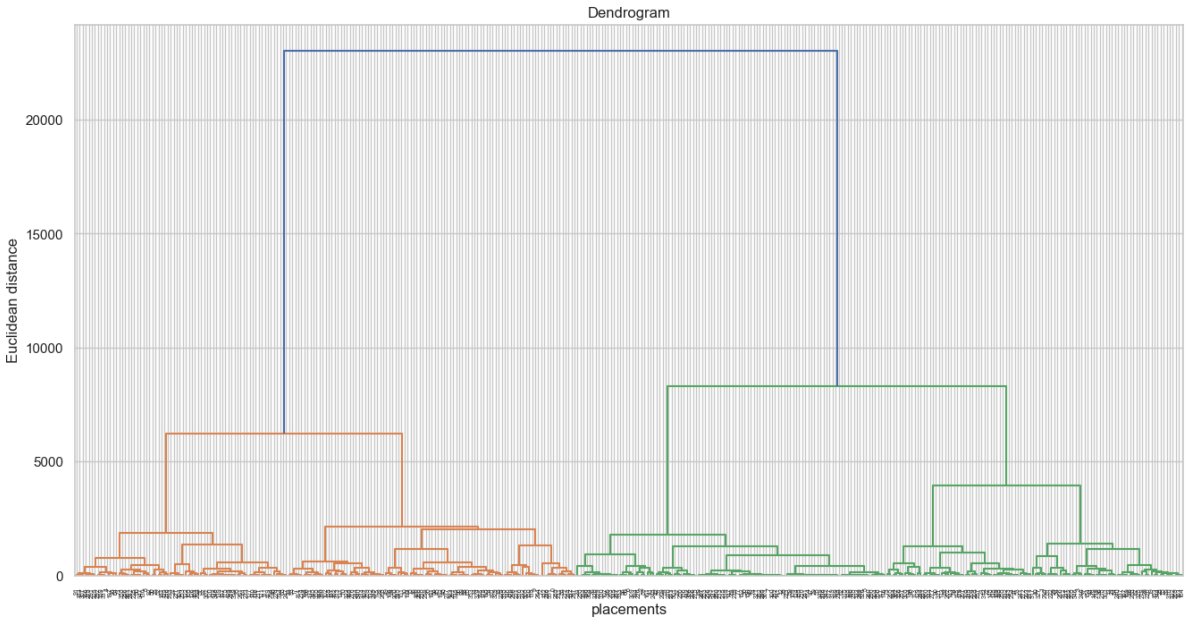
In [617…
```
data_frac.drop('k-m label', axis = 1, inplace = True)
```

In [618…
```
data_frac.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 365 entries, 104455 to 80797
Data columns (total 7 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   company_hash         365 non-null    int32  
 1   ctc                  365 non-null    float64
 2   job_position         365 non-null    int32  
 3   years_of_experience  365 non-null    float64
 4   classs               365 non-null    int64  
 5   designation          365 non-null    int64  
 6   tier                 365 non-null    int64  
dtypes: float64(2), int32(2), int64(3)
memory usage: 20.0 KB
```

In [619…
```
import sys
sys.setrecursionlimit(100000)
```

In [620…
```
# Visual representation of clusters using dendrogram

plt.figure(figsize = (16,8))
import scipy.cluster.hierarchy as sch
dendrogrm = sch.dendrogram(sch.linkage(data_frac, method = 'ward'))
plt.title('Dendrogram')
plt.xlabel('placements')
plt.ylabel('Euclidean distance')
plt.show()
```



In [ ]:

In [621…
```
from sklearn.cluster import AgglomerativeClustering
model = AgglomerativeClustering(n_clusters=3, affinity='euclidean', linkage='ward')
model.fit(data_frac)
```

Out[621]:

AgglomerativeClustering

AgglomerativeClustering(affinity='euclidean', n_clusters=3)

In [622…
```
data_frac['Aglo-label'] = model.fit_predict(data_frac)
```

In [623…
```
data_frac
```

Out[623]:

	company_hash	ctc	job_position	years_of_experience	classs	designation	tier	Aglo labe
104455	2835	0.180180	378	4.0	2	2	3	
13096	1058	0.210210	478	2.0	2	2	2	
75361	186	0.405405	478	10.0	1	2	1	
86532	1200	0.600600	0	6.0	1	1	1	
40848	1716	0.151351	162	6.0	3	2	3	
...	...	...	...	...	...	...	...	
96030	752	0.069069	105	7.0	3	1	3	
9216	1509	0.150150	0	4.0	3	3	3	
142311	644	0.330330	378	8.0	1	2	3	
91327	0	0.219219	230	6.0	3	2	3	
80797	0	0.156156	378	0.0	3	3	3	

365 rows × 8 columns

In [624…
```
final_data
```

Out[624]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	cla
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0	
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0	
2	Others	2015.0	2000000	backend engineer	2020.0	7.0	
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0	
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	5.0	
...	...	...	...	...	...	...	
146048	mvqwrvjo	2011.0	2250000	Others	2019.0	11.0	
146049	vuurt xzw	2008.0	220000	Others	2019.0	14.0	
146050	husqvawgb	2017.0	500000	Others	2020.0	5.0	
146051	vwwgrxnt	2021.0	700000	Others	2021.0	1.0	
146052	bgqsvz onvzrtj	2014.0	1240000	Others	2016.0	8.0	

146053 rows × 10 columns

Comments

1. Above is the final_data with all required features.
2. This data can be submitted to marketing team so that they can focus on those clusters of students who are in dire need of the job.

In [ ]:

In [ ]:

In [ ]: