In [312…]
```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn import preprocessing


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

In [313…]
```
df = pd.read_csv('scaler_clustering.csv')
```

In [314…]
```
df.shape
#shape of data
```

Out[314…]
```
(205843, 7)
```

In [315…]
```
df.head()
```

Out[315…]

	Unnamed: 0	company_hash	email_hash	orgyear	ctc	job_position	ctc_updated_year
0	0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	1100000	Other	2020.0
1	1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018.0	449999	FullStack Engineer	2019.0
2	2	ojzwnvwnxw vx	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2015.0	2000000	Backend Engineer	2020.0
3	3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	2017.0	700000	Backend Engineer	2019.0
4	4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	2017.0	1400000	FullStack Engineer	2019.0

In [316…]
```
#Data types of all attributes
#As we later edit the columns we can conver the object to string
df.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      153281 non-null  object 
 6   ctc_updated_year  205843 non-null  float64
dtypes: float64(2), int64(2), object(3)
memory usage: 11.0+ MB
```

In [317…]
```
#Missing value detection
df.isna().sum()
```

Out[317…]
```
Unnamed: 0             0
company_hash          44
email_hash             0
orgyear               86
ctc                    0
job_position       52562
ctc_updated_year       0
dtype: int64
```

In [318…]
```
#Percentage of missing values in each column

#As we can see job postion has 25% missing values so we need to address this
df.isnull().sum() * 100 / len(df)
```

Out[318…]
```
Unnamed: 0          0.000000
```

```
company_hash         0.021376
email_hash           0.000000
orgyear              0.041779
ctc                  0.000000
job_position        25.534995
ctc_updated_year     0.000000
dtype: float64
```

In [319…]
```
df.nunique()
```

Out[319…]
```
Unnamed: 0          205843
company_hash         37299
email_hash          153443
orgyear                 77
ctc                   3360
job_position          1017
ctc_updated_year         7
dtype: int64
```

In [320…]
```
#There are 37299 unique companies
#There are 153433 unique students
#There are 3360 unique ctc
#Threre are 1017 unique job positions. SDE1 and Software development engineer 1 are counted as different as of nc
```

In [321…]
```
# conversion of categorical attributes to 'category'
# Company hash and job title are categorical variables but we cannot covert them to categorical variables now wit
# some preprocessing as
# 1) Target encoding cannot be done : no target variable
# 2) Ordinal encoding not possible as hashes by themselves have no ordering
# 3) One hot encoding : Not possible as too many unique values and columsn would explode
```

In [322…]
```
#Statistical summary
df.describe()
```

Out[322…]

	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

In [323…]
```
#Value counts
df['company_hash'].value_counts()
```

Out[323…]
```
nvnv wgzohrnvzwj otqcxwto         8337
xzegojo                           5381
vbvkgz                            3481
zgn vuurxwvmrt vwwghzn            3411
wgszxkvzn                         3240
                                  ...
onvqmhwpo                            1
bvsxw ogenfvqt uqxcvnt rxbxnta       1
agsbv ojontbo                        1
vnnhzt xzegwgb                       1
bvptbjnqxu td vbvkgz                 1
Name: company_hash, Length: 37299, dtype: int64
```

In [324…]
```
df['orgyear'].value_counts()
```

Out[324…]
```
2018.0    25256
2019.0    23427
2017.0    23239
2016.0    23043
2015.0    20610
```

```
          ...
2107.0        1
1972.0        1
2101.0        1
208.0         1
200.0         1
Name: orgyear, Length: 77, dtype: int64
```

In [325…
```
df['ctc'].value_counts()
```

Out[325…
```
600000     7832
400000     7598
1000000    7581
500000     7242
800000     6752
           ...
1916000       1
5340000       1
2305000       1
4225000       1
3327000       1
Name: ctc, Length: 3360, dtype: int64
```

In [326…
```
df['job_position'].value_counts()
```

Out[326…
```
Backend Engineer                  43554
FullStack Engineer                24717
Other                             18071
Frontend Engineer                 10417
Engineering Leadership             6870
                                  ...
ayS                                   1
Principal Product Engineer            1
Senior Director of Engineering        1
Seller Support Associate              1
Android Application developer         1
Name: job_position, Length: 1017, dtype: int64
```

In [327…
```
df['ctc_updated_year'].value_counts()
```

Out[327…
```
2019.0    68688
2021.0    64976
2020.0    49444
2017.0     7561
2018.0     6746
2016.0     5501
2015.0     2927
Name: ctc_updated_year, dtype: int64
```

In [328…
```
#Univariate analysis and bivaraite analsis is done after aggregation and feature engineering of the data.
#Otherwse we end up with false / erronous graphical analysis
```

In [329…
```
#2.
df.drop(columns=['email_hash','Unnamed: 0'],inplace=True)
# We are dropping emails as anwyays as emmail has no role in job selection and also emails in case two are preser
# they represent 2 different trajectories of the candidate and can be considered different data points for cluste
# prupose
```

In [330…
```
import re
def remove_special (string):
    new_string=re.sub('[^A-Za-z ]+', '', string)
    return new_string
```

In [331…
```
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[331…
```
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 [332…
```
df['job_position'].nunique()
#number iof unique records alos goes down
```

Out[332…
```
857
```

In [333…
```
df['job_position'].value_counts() / len(df['job_position'])
#23% studetns have jobs as nan
# 50% sudents have job as either nan, backend or full stack engineer
```

Out[333…
```
nan                                0.255350
backend engineer                   0.211588
fullstack engineer                 0.126222
other                              0.087795
frontend engineer                  0.050607
                                     ...
software enginnering specialist    0.000005
android lead                       0.000005
senior analysts                    0.000005
aspnet developer                   0.000005
azure data factory                 0.000005
Name: job_position, Length: 857, dtype: float64
```

In [334…
```
df.drop_duplicates(inplace=True)
df.shape
```

Out[334…
```
(188247, 5)
```

In [335…
```
df['company_hash'].value_counts().sort_index()
```

Out[335…
```
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 [336…
```
df['company_hash'].value_counts() / len(df['company_hash'])
```

Out[336…
```
nvnv wgzohrnvzwj otqcxwto      0.022757
xzegojo                        0.016165
vbvkgz                         0.015963
wgszxkvzn                      0.012021
zgn vuurxwvmrt vwwghzn         0.011735
                                   ...
sggrst                         0.000005
wxznyvbvzx wgbuhntq wytzzvx    0.000005
trtsvzn ntwyzgogen             0.000005
ohwwtoo qtoghqwto              0.000005
bvptbjnqxu td vbvkgz           0.000005
Name: company_hash, Length: 37299, dtype: float64
```

In [337…
```
# As we can see the companies are at which the students work are sparsely distributed. 2.2% is the maximum numbe
# meaning that even for the comapny at which the maximum students work the percentage of student is 2.2%
```

In [338…
```
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[338…
```
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 [339…
```
df['company_hash'].nunique()
#number iof unique records alos goes down
```

Out[339…
```
37208
```

In [340…
```
df['company_hash'].value_counts().sort_index()
```

Out[340…
```
                                            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 [341…
```
print(df.shape)
print(df.drop_duplicates().shape)
df.drop_duplicates(inplace=True)
```

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

In [342…
```
#removing rows where company or job_position is not available
df=df[ ~((df['company_hash']=='') | (df['job_position']==''))]
```

In [343…
```
df.shape
```

Out[343…
```
(188153, 5)
```

In [344…
```
# Filling Null values using Mean Target Inputation for Orgyear
```

In [345…
```
df['orgyear'].isnull().sum()
```

Out[345…
```
86
```

In [346…
```
company_median = df.groupby('company_hash')['orgyear'].median()
```

In [347…
```
company_median
```

Out[347…
```
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 [348…]
```
company_median['a']
```

Out[348…]
```
2017.0
```

In [349…]
```
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 [350…]
```
#Apply the lambda function row wise
df['orgyear']=df.apply(lambda x: null_imputation(company_median, x['company_hash'], x['orgyear']), axis =1)
```

In [351…]
```
len(df[df['orgyear'].isnull()])
```

Out[351…]
```
26
```

In [352…]
```
df=df[~df['orgyear'].isnull()]
```

In [353…]
```
#Significant outlier need to do outlier outlier correction on this
df['orgyear'].value_counts().sort_index()
```

Out[353…]
```
0.0        17
1.0         2
2.0         3
3.0         6
4.0         1
           ..
2101.0      1
2106.0      1
2107.0      1
2204.0      1
20165.0     2
Name: orgyear, Length: 79, dtype: int64
```

In [354…]
```
df['ctc_updated_year'].value_counts().sort_index()
```

Out[354…]
```
2015.0     2896
2016.0     5417
2017.0     7432
2018.0     6656
2019.0    64298
2020.0    45036
2021.0    56392
Name: ctc_updated_year, dtype: int64
```

In [355…]
```
#No outliers in ctc_updates_year
```

In [356…]
```
df.head()
```

Out[356…]

	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 [357…

```
#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)]
```

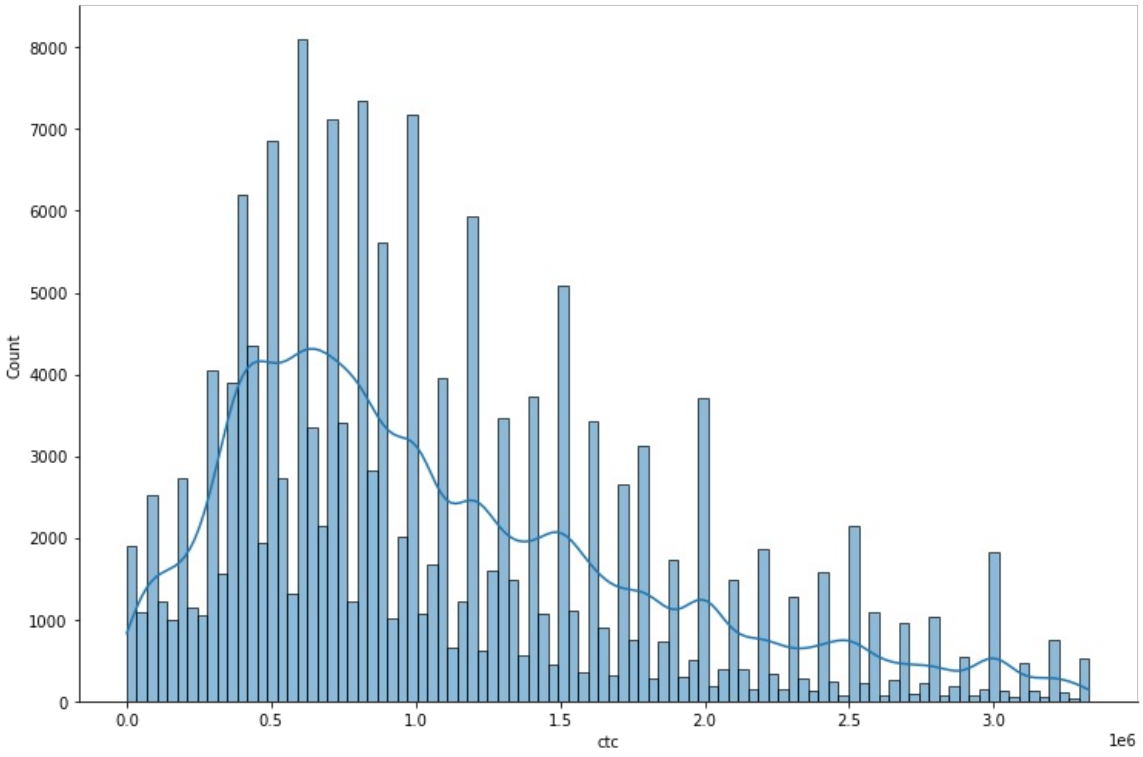
In [358…

```
#Lets do univariate analysis and bivaraite analysis on the numerical columns
```

In [359…

```
sns.displot( x = 'ctc', data = df, kde = True, height = 7, aspect = 1.5)
#Most of the leatners earn between 3 lacs and 20 lacs
```

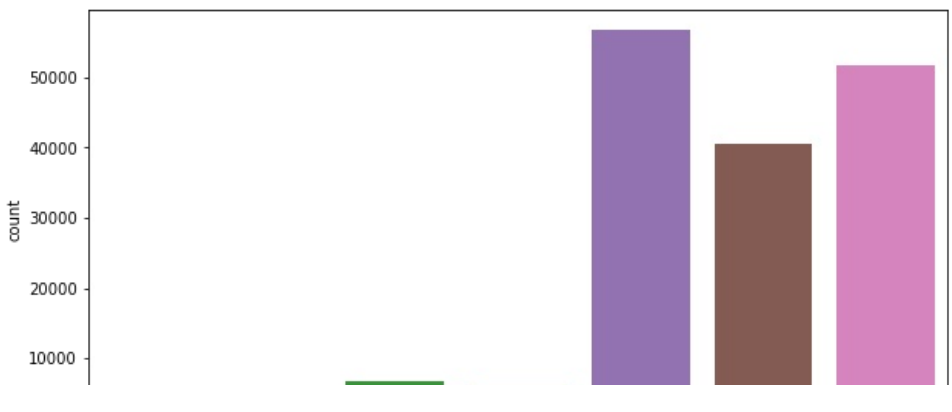
Out[359… <seaborn.axisgrid.FacetGrid at 0x16eeff4ed60>

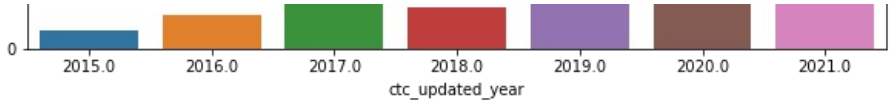


In [360…

```
#ctc_updated_year
fig, ax = plt.subplots(figsize=(10, 5))
sns.countplot(x= 'ctc_updated_year', data = df, ax= ax)
#ctc was updated mostly in the year 2019,2020 and 2021
```

Out[360… <AxesSubplot:xlabel='ctc_updated_year', ylabel='count'>



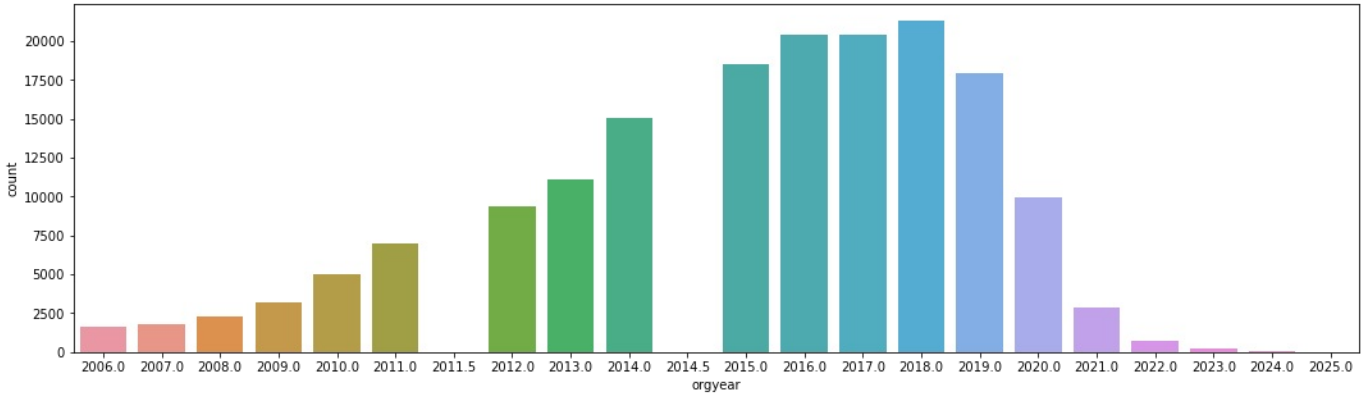


In [361…

```
fig, ax = plt.subplots(figsize=(18, 5))
sns.countplot(x= 'orgyear', data = df, ax= ax)
#Most of the learners joined the companies is between 2014 to 2020 (both inclusive)
```

Out[361…

```
<AxesSubplot:xlabel='orgyear', ylabel='count'>
```



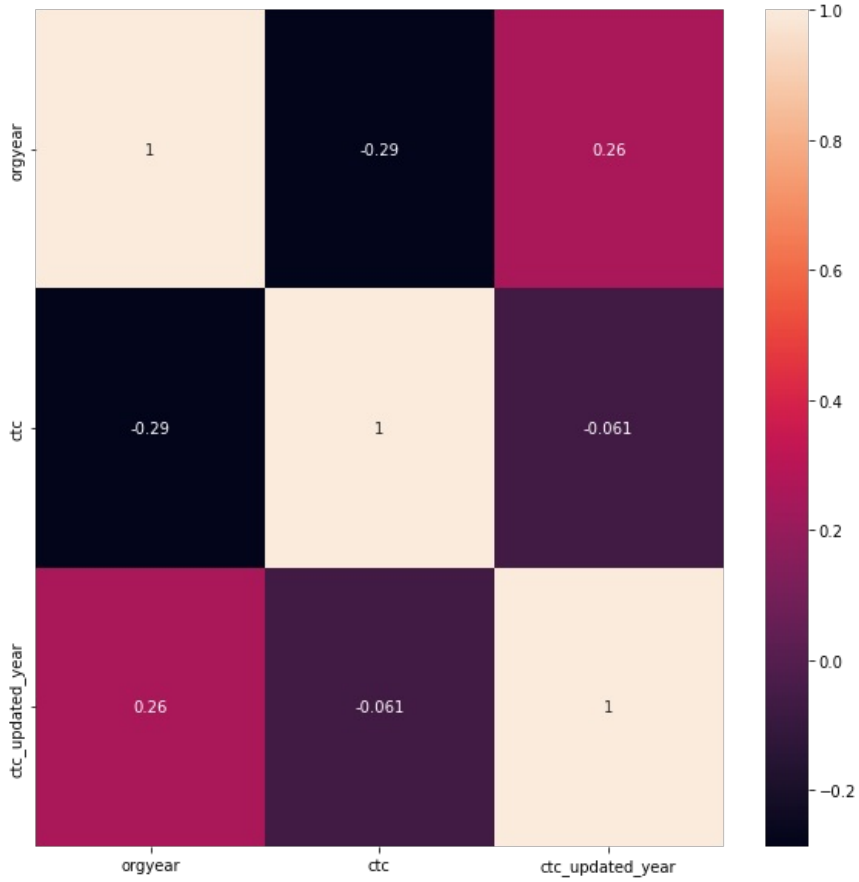
In [362…

```
#Bivariate analysis

#Heatmap
#Since variables aren't normally distributed we do not consider Pearson correlation
fig, ax = plt.subplots(figsize=(10, 10))
Var_Corr = df.corr(method = 'spearman')
sns.heatmap(Var_Corr, xticklabels=Var_Corr.columns, yticklabels=Var_Corr.columns, annot=True, ax=ax)
```

Out[362…

```
<AxesSubplot:>
```



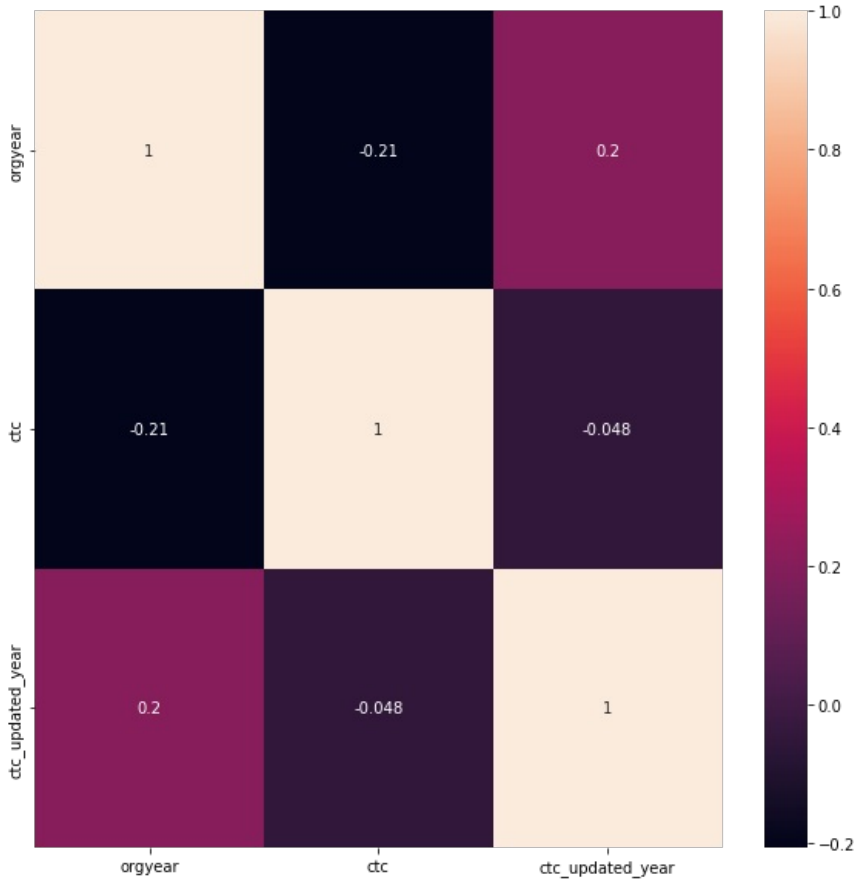
In [363…

```
fig, ax = plt.subplots(figsize=(10, 10))
Var_Corr = df.corr(method = 'kendall')
```

```
sns.heatmap(Var_Corr, xticklabels=Var_Corr.columns, yticklabels=Var_Corr.columns, annot=True, ax=ax)
```

Out[363…

```
<AxesSubplot:>
```



In [364…

```
#Nothing significant from bivariate analysis
```

In [365…

```
print(df.shape)
print(df.drop_duplicates().shape)
df.drop_duplicates(inplace=True)
```

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

In [366…

```
#There are lot of nan's in the job position
```

In [367…

```
#We see some 'nan's in job_position
#Replace string nan in job position with numpy nan
df.loc[df['job_position']=='nan','job_position']=np.nan
```

In [368…

```
# Masking companies by renaming it to "Others" having count less than 5
```

In [369…

```
df.company_hash.value_counts()
```

Out[369…

```
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 [370…

```
(df['company_hash'].value_counts())<=5
```

Out[370…

```
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 [371…

```
df['company_hash'].map(df['company_hash'].value_counts())<=5
```

Out[371…

```
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 [372…

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

Out[372…

	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 [373…

```
df['new']=df['company_hash'].mask(df['company_hash'].map(df['company_hash'].value_counts())<=5, 'Others')
```

In [374…

```
df
```

Out[374…

	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	Others
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 [375…
```
df[df['new']=='Others']['company_hash'].value_counts()
```

Out[375…
```
xzoygqno                  5
bjowyggruvst              5
bjbgztjpvqbv              5
wgznxztznvr vs            5
rtno nqvzougqn            5
                         ..
hmtqnvr                   1
uqgbvn                    1
lp ntwyzgogen rna zgxav   1
sggrst                    1
wyvqntq wgbbhzxwvnxgzo    1
Name: company_hash, Length: 31066, dtype: int64
```

In [376…
```
#By deafult axis = 0, means we are going column wise
df=df.apply(lambda x: x.mask(x.map(x.value_counts())<=5,'Others') if x.name=='company_hash' else x)
```

In [377…
```
df['company_hash'].value_counts()
```

Out[377…
```
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 [378…
```
df.drop(columns='new',inplace=True)
```

In [379…
```
#Creating Years of Experience Columns
```

In [380…
```
df.drop_duplicates(inplace=True)
df.shape
```

Out[380…
```
(147140, 5)
```

In [381…
```
#orgyear check
df['orgyear'] = df.apply(lambda x: x['orgyear'] if x['orgyear'] <=2022 else 2022, axis=1)
```

In [382…
```
df['years_of_experience']=2022-df['orgyear']
```

In [383…
```
#ctc_updated_year_check
df['ctc_updated_year'] = df.apply(lambda x: x['orgyear'] if x['ctc_updated_year'] < x['orgyear']
                                                         else x['ctc_updated_year'], axis=1)
```

In [384…
```
#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 [385…
```
df.isnull().sum()
#All good now
```

Out[385…
```
company_hash          0
orgyear               0
ctc                   0
```

```
job_position           0
ctc_updated_year       0
years_of_experience    0
dtype: int64
```

In [386…
```
df.describe()
```

Out[386…

	orgyear	ctc	ctc_updated_year	years_of_experience
count	147140.000000	1.471400e+05	147140.000000	147140.000000
mean	2015.481647	1.127801e+06	2019.600041	6.518353
std	3.311169	7.435970e+05	1.339111	3.311169
min	2006.000000	2.000000e+00	2015.000000	0.000000
25%	2013.000000	5.700000e+05	2019.000000	4.000000
50%	2016.000000	9.500000e+05	2020.000000	6.000000
75%	2018.000000	1.550000e+06	2021.000000	9.000000
max	2022.000000	3.330000e+06	2022.000000	16.000000

In [387…
```
#Manual clustering
```

In [388…
```
# company, job position and years of experience
```

In [389…
```
grp_cjy =  df.groupby(['years_of_experience', 'job_position','company_hash'], as_index=True)['ctc'].describe()
```

In [390…
```
grp_cjy
```

Out[390…

			count	mean	std	min	25%	50%	75%	max
years_of_experience	**job_position**	**company_hash**								
0.0	**Others**	**Others**	54.0	7.421518e+05	651538.866897	200.0	282500.0	525000.0	1154999.75	3000000.0
		agzn fgqp xz vzj gqsvzxkvnxgz	1.0	1.600000e+06	NaN	1600000.0	1600000.0	1600000.0	1600000.00	1600000.0
		atrgxnnt	1.0	1.000000e+06	NaN	1000000.0	1000000.0	1000000.0	1000000.00	1000000.0
		atrr	1.0	1.000000e+06	NaN	1000000.0	1000000.0	1000000.0	1000000.00	1000000.0
		atrr ntwyzgrgsxto	2.0	1.000000e+06	282842.712475	800000.0	900000.0	1000000.0	1100000.00	1200000.0
...	**...**	**...**	...	...	...	...	...	...	...	...
16.0	**support engineer**	**xzegojo**	1.0	8.000000e+05	NaN	800000.0	800000.0	800000.0	800000.00	800000.0
		xzegq	1.0	9.000000e+05	NaN	900000.0	900000.0	900000.0	900000.00	900000.0
		ywr ntwyzgrgsxto	2.0	8.500000e+05	494974.746831	500000.0	675000.0	850000.0	1025000.00	1200000.0
		zvz	1.0	4.000000e+05	NaN	400000.0	400000.0	400000.0	400000.00	400000.0
	team lead	**utqoxontzn ojontbo**	1.0	1.600000e+06	NaN	1600000.0	1600000.0	1600000.0	1600000.00	1600000.0

56097 rows × 8 columns

In [391…
```
#Describe returns a dataframe
type(grp_cjy)
```

Out[391…
```
pandas.core.frame.DataFrame
```

In [392…
```
df_cjy=df.merge(grp_cjy, on=['years_of_experience', 'job_position','company_hash'], how = 'left')
```

In [393…
```
df_cjy.head()
```

Out[393…

company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	count	mean	std	min

0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0	1.0	1.100000e+06	NaN	1100000.0	1100
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0	7.0	7.742856e+05	250922.324350	449999.0	610
2	Others	2015.0	2000000	backend engineer	2020.0	7.0	456.0	9.609559e+05	776546.830662	1000.0	307
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0	7.0	1.158571e+06	404780.951933	700000.0	825
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	5.0	1.0	1.400000e+06	NaN	1400000.0	1400

In [394…
```
df_cjy.sort_values(['years_of_experience','job_position','company_hash'])
```

Out[394…

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	count	mean	std	min
896	Others	2022.0	120000	Others	2022.0	0.0	54.0	7.421518e+05	651538.866897	200.0
2600	Others	2022.0	430000	Others	2022.0	0.0	54.0	7.421518e+05	651538.866897	200.0
7701	Others	2022.0	570000	Others	2022.0	0.0	54.0	7.421518e+05	651538.866897	200.0
7880	Others	2022.0	550000	Others	2022.0	0.0	54.0	7.421518e+05	651538.866897	200.0
8804	Others	2022.0	680000	Others	2022.0	0.0	54.0	7.421518e+05	651538.866897	200.0
...	...	...	...	...	...	...	...	...	...	...
73964	xzegq	2006.0	900000	support engineer	2021.0	16.0	1.0	9.000000e+05	NaN	900000.0
11374	ywr ntwyzgrgsxto	2006.0	500000	support engineer	2021.0	16.0	2.0	8.500000e+05	494974.746831	500000.0
37281	ywr ntwyzgrgsxto	2006.0	1200000	support engineer	2021.0	16.0	2.0	8.500000e+05	494974.746831	500000.0
14290	zvz	2006.0	400000	support engineer	2021.0	16.0	1.0	4.000000e+05	NaN	400000.0
59896	utqoxontzn ojontbo	2006.0	1600000	team lead	2021.0	16.0	1.0	1.600000e+06	NaN	1600000.0

147140 rows × 14 columns

In [395…
```
df_cjy.drop_duplicates(inplace=True)
df_cjy.shape
#no change till now
```

Out[395… (146053, 14)

In [396…
```
#Need to add designation based on how much each employee earns

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 [397…
```
df_cjy['designation'] =df_cjy.apply(lambda x: condition_designation(x['ctc'],x['50%'],x['75%']), axis=1)
```

In [398…
```
df_cjy.head()
```

Out[398…

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	count	mean	std	min	
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0	1.0	1.100000e+06	NaN	1100000.0	1100
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0	7.0	7.742856e+05	250922.324350	449999.0	610
2	Others	2015.0	2000000	backend engineer	2020.0	7.0	456.0	9.609559e+05	776546.830662	1000.0	307
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0	7.0	1.158571e+06	404780.951933	700000.0	825
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	5.0	1.0	1.400000e+06	NaN	1400000.0	1400

In [399…
```
df_cjy['designation'].value_counts(normalize=True)*100
```

Out[399…
```
2    44.129186
3    34.167734
1    21.703080
Name: designation, dtype: float64
```

In [400…
```
# Manual Clustering based on company and job position
```

In [401…
```
grouped_c_j=df.groupby(['company_hash', 'job_position'])['ctc'].describe()
```

In [402…
```
grouped_c_j
```

Out[402…

company_hash	job_position	count	mean	std	min	25%	50%	75%	max
Others	Others	3296.0	1.021510e+06	832031.028338	15.0	360000.0	800000.0	1500000.0	3327000.0
	a group chat application	1.0	5.000000e+05	NaN	500000.0	500000.0	500000.0	500000.0	500000.0
	abap developer	1.0	5.000000e+05	NaN	500000.0	500000.0	500000.0	500000.0	500000.0
	administrative clerk	1.0	5.000000e+05	NaN	500000.0	500000.0	500000.0	500000.0	500000.0
	administrator	1.0	3.800000e+05	NaN	380000.0	380000.0	380000.0	380000.0	380000.0
...	...	...	...	...	...	...	...	...	...
zxztrtvuo	fullstack engineer	7.0	8.725714e+05	362166.562575	500000.0	637500.0	710000.0	1061500.0	1500000.0
	ios engineer	1.0	1.200000e+06	NaN	1200000.0	1200000.0	1200000.0	1200000.0	1200000.0
	member of technical staff at nineleaps	1.0	1.200000e+06	NaN	1200000.0	1200000.0	1200000.0	1200000.0	1200000.0
	other	2.0	4.500000e+05	0.000000	450000.0	450000.0	450000.0	450000.0	450000.0
	software developer intern	1.0	1.200000e+06	NaN	1200000.0	1200000.0	1200000.0	1200000.0	1200000.0

21596 rows × 8 columns

In [403…
```
df.drop_duplicates().shape
```

Out[403…
```
(146053, 6)
```

In [404…
```
df_cj=df.merge(grouped_c_j, on=['company_hash', 'job_position'], how='left')
```

In [405…
```
df_cj
```

Out[405…

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	count	mean	std	min
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0	2.0	1.085000e+06	2.121320e+04	1070000.0
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0	25.0	9.882000e+05	4.874998e+05	300000.0
2	Others	2015.0	2000000	backend engineer	2020.0	7.0	3927.0	9.958526e+05	8.105078e+05	1000.0
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0	24.0	1.416667e+06	5.453413e+05	520000.0
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	5.0	3.0	8.466667e+05	4.801389e+05	540000.0
...	...	...	...	...	...	...	...	...	...	...
147135	mvqwrvjo	2011.0	2250000	Others	2019.0	11.0	64.0	1.259969e+06	5.777488e+05	500000.0
147136	vuurt xzw	2008.0	220000	Others	2019.0	14.0	16.0	1.568312e+06	1.231984e+06	60000.0
147137	husqvawgb	2017.0	500000	Others	2020.0	5.0	13.0	1.000769e+06	3.300369e+05	500000.0
147138	vwwgrxnt	2021.0	700000	Others	2021.0	1.0	35.0	1.200371e+06	5.635221e+05	300000.0
147139	bgqsvz onvzrtj	2014.0	1240000	Others	2016.0	8.0	105.0	1.801581e+06	6.903383e+05	100000.0

147140 rows × 14 columns

In [406…
```
df_cj.sort_values(['company_hash','job_position'])
```

Out[406…

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	count	mean	std	min
9	Others	2019.0	360000	Others	2019.0	3.0	3296.0	1.021510e+06	832031.028338	15.0
71	Others	2021.0	2200000	Others	2021.0	1.0	3296.0	1.021510e+06	832031.028338	15.0
72	Others	2020.0	300000	Others	2020.0	2.0	3296.0	1.021510e+06	832031.028338	15.0
87	Others	2009.0	350000	Others	2021.0	13.0	3296.0	1.021510e+06	832031.028338	15.0
135	Others	2017.0	500000	Others	2020.0	5.0	3296.0	1.021510e+06	832031.028338	15.0
...	...	...	...	...	...	...	...	...	...	...
122932	zxztrtvuo	2013.0	1200000	ios engineer	2017.0	9.0	1.0	1.200000e+06	NaN	1200000.0
53944	zxztrtvuo	2016.0	1200000	member of technical staff at nineleaps	2020.0	6.0	1.0	1.200000e+06	NaN	1200000.0
9204	zxztrtvuo	2020.0	450000	other	2020.0	2.0	2.0	4.500000e+05	0.000000	450000.0
134141	zxztrtvuo	2019.0	450000	other	2020.0	3.0	2.0	4.500000e+05	0.000000	450000.0
37362	zxztrtvuo	2016.0	1200000	software developer intern	2020.0	6.0	1.0	1.200000e+06	NaN	1200000.0

147140 rows × 14 columns

In [407…
```
df_cj.drop_duplicates(inplace=True)
df_cj.shape
```

Out[407…
```
(146053, 14)
```

In [408…
```
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 [409…
```
df_cj['class'] =df_cj.apply(lambda x: condition_classs(x['ctc'],x['50%'],x['75%']), axis=1)
```

In [410…
```
df_cj
```

Out[410…

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	count	mean	std	min
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0	2.0	1.085000e+06	2.121320e+04	1070000.0
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0	25.0	9.882000e+05	4.874998e+05	300000.0
2	Others	2015.0	2000000	backend engineer	2020.0	7.0	3927.0	9.958526e+05	8.105078e+05	1000.0
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0	24.0	1.416667e+06	5.453413e+05	520000.0
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	5.0	3.0	8.466667e+05	4.801389e+05	540000.0
...	...	...	...	...	...	...	...	...	...	...
147135	mvqwrvjo	2011.0	2250000	Others	2019.0	11.0	64.0	1.259969e+06	5.777488e+05	500000.0
147136	vuurt xzw	2008.0	220000	Others	2019.0	14.0	16.0	1.568312e+06	1.231984e+06	60000.0
147137	husqvawgb	2017.0	500000	Others	2020.0	5.0	13.0	1.000769e+06	3.300369e+05	500000.0
147138	vwwgrxnt	2021.0	700000	Others	2021.0	1.0	35.0	1.200371e+06	5.635221e+05	300000.0
147139	bgqsvz onvzrtj	2014.0	1240000	Others	2016.0	8.0	105.0	1.801581e+06	6.903383e+05	100000.0

146053 rows × 15 columns

In [411…
```
df_cj['class'].value_counts(normalize=True)*100
```

Out[411…
```
3    43.689620
2    31.861037
1    24.449344
Name: class, dtype: float64
```

In [412…
```
df_cj[df_cj['class']==1][['job_position','ctc']]
```

Out[412…

	job_position	ctc
0	other	1100000
2	backend engineer	2000000
4	fullstack engineer	1400000
15	backend engineer	2030000
17	Others	1400000
...	...	...
147124	Others	1330000
147125	Others	2100000
147126	Others	1800000
147134	Others	2280000
147135	Others	2250000

35709 rows × 2 columns

In [413…
```
# job position that has the highest class
df_cj[df_cj['class']==1][['job_position','ctc']].groupby('job_position')['ctc'].describe()
```

Out[413…

job_position	count	mean	std	min	25%	50%	75%	max
Others	8229.0	1.930377e+06	695369.034053	100000.0	1400000.0	1900000.0	2500000.0	3330000.0
android engineer	913.0	1.784897e+06	638704.770985	14000.0	1320000.0	1700000.0	2200000.0	3300000.0
application developer	1.0	1.150000e+06	NaN	1150000.0	1150000.0	1150000.0	1150000.0	1150000.0
application developer analyst	1.0	6.000000e+05	NaN	600000.0	600000.0	600000.0	600000.0	600000.0
application development analyst	2.0	8.150000e+05	233345.237792	650000.0	732500.0	815000.0	897500.0	980000.0
...	...	...	...	...	...	...	...	...
support engineer	683.0	1.190779e+06	552019.578789	350000.0	830000.0	1000000.0	1400000.0	3310000.0
system engineer	10.0	8.420000e+05	373118.986086	400000.0	550000.0	775000.0	1100000.0	1500000.0
teaching assistant	1.0	1.800000e+06	NaN	1800000.0	1800000.0	1800000.0	1800000.0	1800000.0
team lead	2.0	1.800000e+06	565685.424949	1400000.0	1600000.0	1800000.0	2000000.0	2200000.0
technology analyst	3.0	8.966667e+05	351046.055858	660000.0	695000.0	730000.0	1015000.0	1300000.0

108 rows × 8 columns

In [414…
```
df_cj.head()
```

Out[414…

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	count	mean	std	min	
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0	2.0	1.085000e+06	21213.203436	1070000.0	107
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0	25.0	9.882000e+05	487499.789590	300000.0	60
2	Others	2015.0	2000000	backend engineer	2020.0	7.0	3927.0	9.958526e+05	810507.825376	1000.0	30
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0	24.0	1.416667e+06	545341.270627	520000.0	104
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	5.0	3.0	8.466667e+05	480138.868801	540000.0	57

In [415…]
```
df_cjy.head()
```

Out[415…]

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	count	mean	std	min	
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0	1.0	1.100000e+06	NaN	1100000.0	1100
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0	7.0	7.742856e+05	250922.324350	449999.0	610
2	Others	2015.0	2000000	backend engineer	2020.0	7.0	456.0	9.609559e+05	776546.830662	1000.0	307
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0	7.0	1.158571e+06	404780.951933	700000.0	825
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	5.0	1.0	1.400000e+06	NaN	1400000.0	1400

In [416…]
```
df_cj.shape
```

Out[416…]
```
(146053, 15)
```

In [417…]
```
df_cjy.shape
```

Out[417…]
```
(146053, 15)
```

In [418…]
```
df_cj.drop(columns=['count','mean','std','min','25%','50%','75%','max'],inplace=True)
df_cjy.drop(columns=['count','mean','std','min','25%','50%','75%','max'],inplace=True)
```

In [419…]
```
df_cj.drop_duplicates().shape
```

Out[419…]
```
(146053, 7)
```

In [420…]
```
df_cjy.drop_duplicates().shape
```

Out[420…]
```
(146053, 7)
```

In [421…]
```
df_cjy_cj=df_cj.merge(df_cjy, on=['company_hash','orgyear','ctc','job_position','ctc_updated_year','years_of_expe
```

In [422…]
```
df_cjy_cj.shape
```

Out[422…]
```
(146053, 8)
```

In [423…]
```
df_cjy_cj.drop_duplicates().shape
```

Out[423…]
```
(146053, 8)
```

In [424…]
```
#Manual clustering based on company
```

In [425…]
```
grouped_c=df.groupby(['company_hash'])['ctc'].describe()
```

In [426…]
```
df_c=df.merge(grouped_c, on=['company_hash'], how='left')
```

In [427…]
```
df_c.head(5)
```

Out[427…]

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	count	mean	std	min

0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0	9.0	1.115667e+06	458111.885897	500000.0	80(
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0	384.0	1.055291e+06	636095.670307	10000.0	60(
2	Others	2015.0	2000000	backend engineer	2020.0	7.0	24903.0	9.647442e+05	760124.617272	15.0	39(
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0	59.0	1.455508e+06	655423.458086	200000.0	107
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	5.0	6.0	9.400000e+05	389871.773792	540000.0	625

In [428…

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

Out[428…

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	count	mean	std	mi
30026	Others	2017.0	730000	ios engineer	2020.0	5.0	24903.0	964744.216801	760124.617272	15.(
37763	Others	2021.0	1200000	Others	2021.0	1.0	24903.0	964744.216801	760124.617272	15.(
76928	Others	2022.0	1410000	Others	2022.0	0.0	24903.0	964744.216801	760124.617272	15.(
17694	Others	2012.0	900000	fullstack engineer	2020.0	10.0	24903.0	964744.216801	760124.617272	15.(
76939	Others	2019.0	1900000	Others	2021.0	3.0	24903.0	964744.216801	760124.617272	15.(
...	...	...	...	...	...	...	...	...	...	..
73909	zxztrtvuo	2020.0	400000	Others	2020.0	2.0	69.0	957217.376812	564608.271459	400000.(
121923	zxztrtvuo	2015.0	1200000	frontend engineer	2019.0	7.0	69.0	957217.376812	564608.271459	400000.(
81725	zxztrtvuo	2019.0	400000	Others	2021.0	3.0	69.0	957217.376812	564608.271459	400000.(
116629	zxztrtvuo	2017.0	1000000	backend engineer	2019.0	5.0	69.0	957217.376812	564608.271459	400000.(
121737	zxztrtvuo	2020.0	450000	Others	2020.0	2.0	69.0	957217.376812	564608.271459	400000.(

147140 rows × 14 columns

In [429…

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

```
(147140, 14)
(146053, 14)
```

In [430…

```
# Adding Tier based on salary in each company
```

In [431…

```
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 [432…

```
df_c['tier'] =df_c.apply(lambda x: condition_tier(x['ctc'],x['50%'],x['75%']),axis=1)
```

In [433…

```
df_c.head()
```

Out[433…

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	count	mean	std	min
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0	9.0	1.115667e+06	458111.885897	500000.0	80(
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0	384.0	1.055291e+06	636095.670307	10000.0	60(
2	Others	2015.0	2000000	backend engineer	2020.0	7.0	24903.0	9.647442e+05	760124.617272	15.0	39(
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0	59.0	1.455508e+06	655423.458086	200000.0	107
				fullstack							

4	qxen sqghu	2017.0	1400000	engineer	2019.0	5.0	6.0	9.400000e+05	389871.773792	540000.0	625

In [434…
```
df_c['tier'].value_counts(normalize=True)*100
```

Out[434…
```
3    47.974718
2    28.165013
1    23.860269
Name: tier, dtype: float64
```

In [435…
```
df_cjy_cj_c=df_cjy_cj.merge(df_c, on=['company_hash','orgyear','ctc','job_position','ctc_updated_year','years_of_
```

In [436…
```
df_cjy_cj_c.head(10)
```

Out[436…

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	class	designation	count	mean	
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0	1	2	9.0	1.115667e+06	458111.885
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0	3	3	384.0	1.055291e+06	636095.67(
2	Others	2015.0	2000000	backend engineer	2020.0	7.0	1	1	24903.0	9.647442e+05	760124.617
3	ngpgutaxv	2017.0	700000	backend engineer	2019.0	5.0	3	3	59.0	1.455508e+06	655423.458
4	qxen sqghu	2017.0	1400000	fullstack engineer	2019.0	5.0	1	2	6.0	9.400000e+05	389871.773
5	yvuuxrj hzbvqqxta bvqptnxzs ucn rna	2018.0	700000	fullstack engineer	2020.0	4.0	2	2	6.0	9.066667e+05	539728.32(
6	lubgqsvz wyvot wg	2018.0	1500000	fullstack engineer	2019.0	4.0	3	3	863.0	1.704393e+06	676089.47
7	vwwtznhqt ntwyzgrgsj	2019.0	400000	backend engineer	2019.0	3.0	3	3	24.0	6.633333e+05	265782.956
8	utqoxontzn ojontbo	2020.0	450000	Others	2020.0	2.0	3	2	415.0	9.752988e+05	555073.324
9	utqoxontzn ojontbo	2020.0	450000	Others	2020.0	2.0	3	2	415.0	9.752988e+05	555073.324

In [437…
```
df_cjy_cj_c.drop(columns=['count','mean','std','min','25%','50%','75%','max'],inplace=True)
```

In [438…
```
df_cjy_cj_c.shape
```

Out[438…
```
(147140, 9)
```

In [439…
```
data=df_cjy_cj_c.copy(deep=True)
```

In [440…
```
data.shape
```

Out[440…
```
(147140, 9)
```

In [441…
```
data.head(10)
```

Out[441…

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

6	lubgqsvz wyvot wg	2018.0	1500000	fullstack engineer	2019.0	4.0	3	3	3
7	vwwtznhqt ntwyzgrgsj	2019.0	400000	backend engineer	2019.0	3.0	3	3	3
8	utqoxontzn ojontbo	2020.0	450000	Others	2020.0	2.0	3	2	3
9	utqoxontzn ojontbo	2020.0	450000	Others	2020.0	2.0	3	2	3

In [ ]:

In [442…

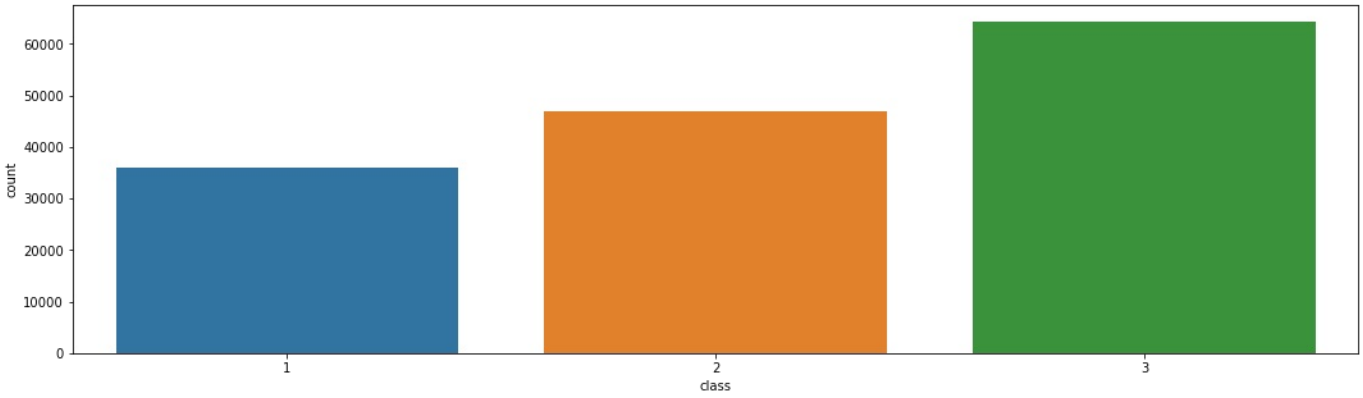
```
#Univariate analysis of the vairables added

#Class
#Most are in class 2 and 3

fig, ax = plt.subplots(figsize=(18, 5))
sns.countplot(x= 'class', data = data, ax= ax)
```

Out[442…

```
<AxesSubplot:xlabel='class', ylabel='count'>
```



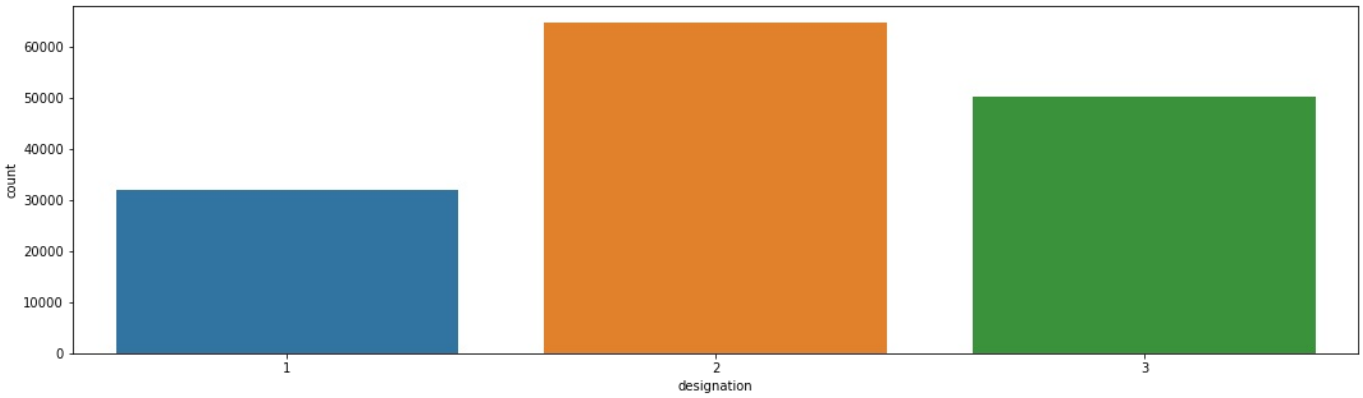
In [443…

```
#Designation
#Most have designation 2

fig, ax = plt.subplots(figsize=(18, 5))
sns.countplot(x= 'designation', data = data, ax= ax)
```

Out[443…

```
<AxesSubplot:xlabel='designation', ylabel='count'>
```



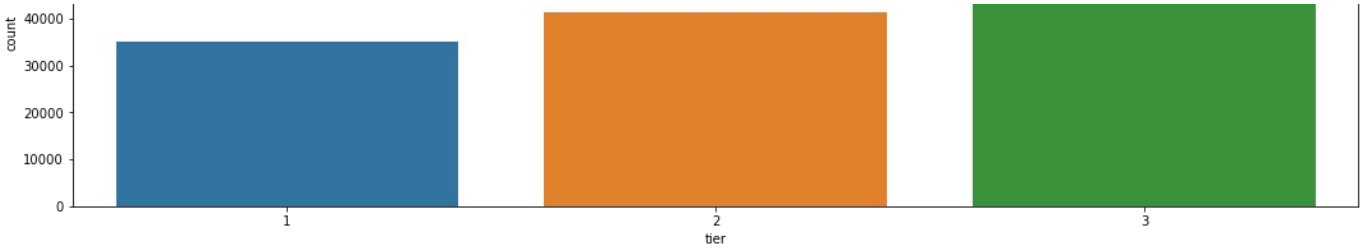
In [444…

```
#Tier
# Tier 3 , 2, 1 and in decreasing order

fig, ax = plt.subplots(figsize=(18, 5))
sns.countplot(x= 'tier', data = data, ax= ax)
```

Out[444…

```
<AxesSubplot:xlabel='tier', ylabel='count'>
```



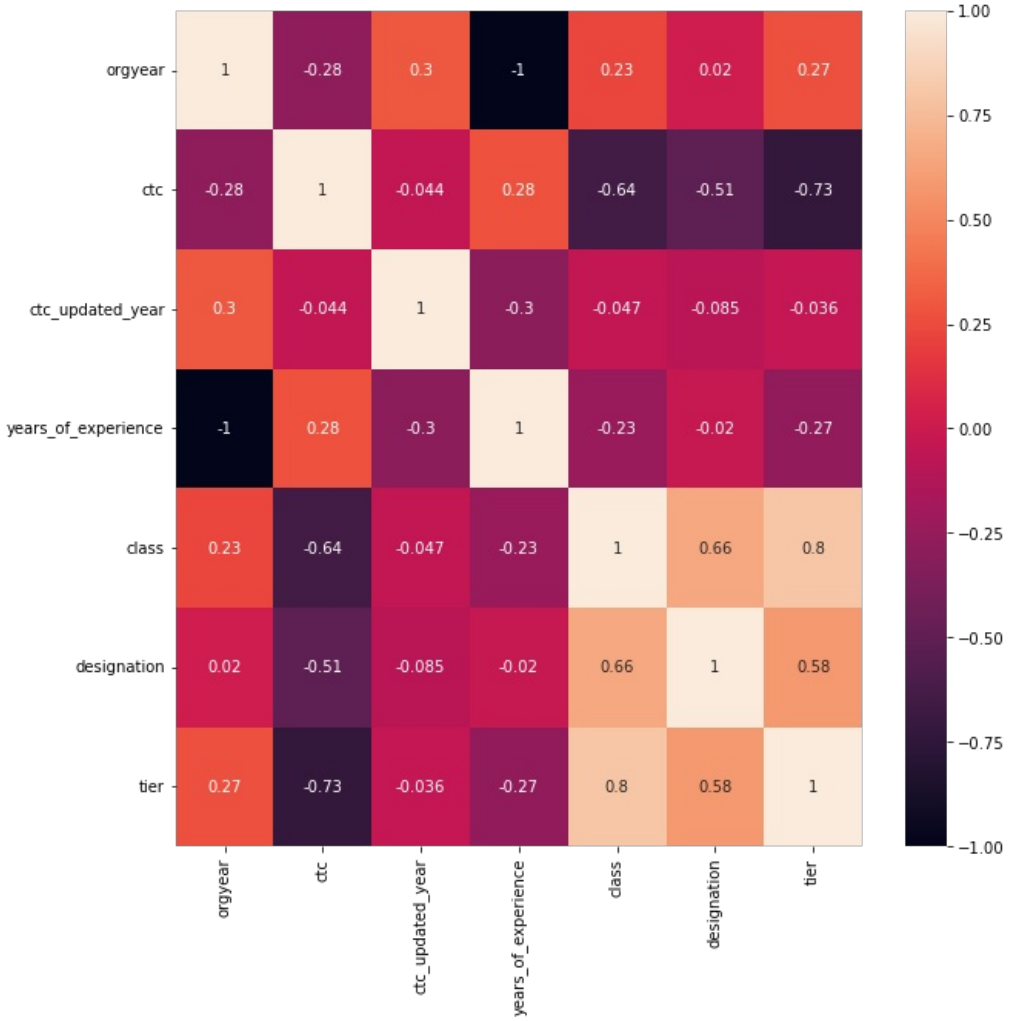
In [445…

```
#Bivarate analysis after new variables are added

fig, ax = plt.subplots(figsize=(10, 10))
Var_Corr = data.corr(method = 'spearman')
sns.heatmap(Var_Corr, xticklabels=Var_Corr.columns, yticklabels=Var_Corr.columns, annot=True, ax=ax)
```

Out[445…

```
<AxesSubplot:>
```



In [ ]:

```
#1) Designation and tier are moderately correlated
#2) Class and tier are strongly correlated
#3) Designation and class are strogly correlated
#4) ctc is inversely corrrelated to both class and tier
```

In [163…

```
#Unsupervised Learning
```

In [164…

```
#KMeans
#1. Check Clustering Tendency
#2. Do elbow method etc for checking the number of optimal clusters (bring it to 3)
#3. Do kmeans clustering
#4. Hierarchial clustering of sample dataset (shuffle and take 50k rows
```

In [165…

```
#We can use simple label encoding as well for company_hash and job_position

#For company_hash we replace each value with average ctc for that company
```

```
dss = data.groupby('company_hash')['ctc'].mean()
```

In [166…
```
data['company_hash'] = data['company_hash'].map(dss)
```

In [167…
```
data.head()
```

Out[167…

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	class	designation	tier
0	1.115667e+06	2016.0	1100000	other	2020.0	6.0	1	2	2
1	1.055291e+06	2018.0	449999	fullstack engineer	2019.0	4.0	3	3	3
2	9.647442e+05	2015.0	2000000	backend engineer	2020.0	7.0	1	1	1
3	1.455508e+06	2017.0	700000	backend engineer	2019.0	5.0	3	3	3
4	9.400000e+05	2017.0	1400000	fullstack engineer	2019.0	5.0	1	2	1

In [168…
```
jss = data.groupby('job_position')['years_of_experience'].mean()
```

In [169…
```
data['job_position'] = data['job_position'].map(jss)
```

In [170…
```
data.head()
```

Out[170…

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	class	designation	tier
0	1.115667e+06	2016.0	1100000	5.953188	2020.0	6.0	1	2	2
1	1.055291e+06	2018.0	449999	6.039619	2019.0	4.0	3	3	3
2	9.647442e+05	2015.0	2000000	6.707304	2020.0	7.0	1	1	1
3	1.455508e+06	2017.0	700000	6.707304	2019.0	5.0	3	3	3
4	9.400000e+05	2017.0	1400000	6.039619	2019.0	5.0	1	2	1

In [171…
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 147140 entries, 0 to 147139
Data columns (total 9 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   company_hash         147140 non-null  float64
 1   orgyear              147140 non-null  float64
 2   ctc                  147140 non-null  int64  
 3   job_position         147140 non-null  float64
 4   ctc_updated_year     147140 non-null  float64
 5   years_of_experience  147140 non-null  float64
 6   class                147140 non-null  int64  
 7   designation          147140 non-null  int64  
 8   tier                 147140 non-null  int64  
dtypes: float64(5), int64(4)
memory usage: 11.2 MB
```

In [172…
```
data.drop(columns=['orgyear'],inplace=True)
data.drop(columns=['ctc_updated_year'],inplace=True)
```

In [173…
```
data.isnull().sum()
```

Out[173…
```
company_hash           0
ctc                    0
job_position           0
years_of_experience    0
class                  0
designation            0
tier                   0
dtype: int64
```

In [174…
```
data.columns
```

Out[174…
```
Index(['company_hash', 'ctc', 'job_position', 'years_of_experience', 'class',
       'designation', 'tier'],
      dtype='object')
```

In [182…
```
from sklearn.preprocessing import MinMaxScaler
ms = MinMaxScaler()
# ------- fix X #do this after label encoding
#I tried multiple combinations with caling just company_hash abd job_poistion as well but then also hopkinds scor
# very low
data[data.columns] = ms.fit_transform(data[data.columns])
```

In [183…
```
print(data.head())
```

```
   company_hash       ctc  job_position  years_of_experience  class  \
0      0.361399  0.330330      0.372074               0.3750    0.0
1      0.339628  0.135134      0.377476               0.2500    1.0
2      0.306978  0.600600      0.419206               0.4375    0.0
3      0.483942  0.210210      0.419206               0.3125    1.0
4      0.298056  0.420420      0.377476               0.3125    0.0

   designation  tier
0          0.5   0.5
1          1.0   1.0
2          0.0   0.0
3          1.0   1.0
4          0.5   0.0
```

In [177…
```
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn.cluster import KMeans
from pyclustertend import hopkins
from sklearn.preprocessing import scale
```

In [178…
```
data.dropna(inplace=True)
```

In [179…
```
hop=hopkins(data,150)
```

In [184…
```
print(hop)
```

```
0.03332337564844248
```

In [279…
```
#hop value of less than 0.5 means not clusterable
#Lets try label encoding on company hash and job position
```

In [280…
```
data1=df_cjy_cj_c.copy(deep=True)
```

In [281…
```
label_encoder = preprocessing.LabelEncoder()
data1['company_hash']= label_encoder.fit_transform(data1['company_hash'])
data1['job_position']= label_encoder.fit_transform(data1['job_position'])
```

In [282…
```
data1.head()
```

Out[282…

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience	class	designation	tier
0	45	2016.0	1100000	378	2020.0	6.0	1	2	2
1	1497	2018.0	449999	235	2019.0	4.0	3	3	3
2	0	2015.0	2000000	105	2020.0	7.0	1	1	1
3	936	2017.0	700000	105	2019.0	5.0	3	3	3
4	1535	2017.0	1400000	235	2019.0	5.0	1	2	1

In [283…
```
data1.drop(columns=['orgyear'],inplace=True)
data1.drop(columns=['ctc_updated_year'],inplace=True)
```

In [284…

```
data1[['ctc']] = ms.fit_transform(data1[['ctc']])
print(data1.head())
```

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

   designation  tier
0            2     2
1            3     3
2            1     1
3            3     3
4            2     1
```

In [285…

```
data1.dropna(inplace=True)
```

In [286…

```
data2 = data1.copy(deep=True)
```

In [287…

```
hop=hopkins(data1,150)
```

In [288…

```
print(hop)
```

```
0.05438168263179502
```

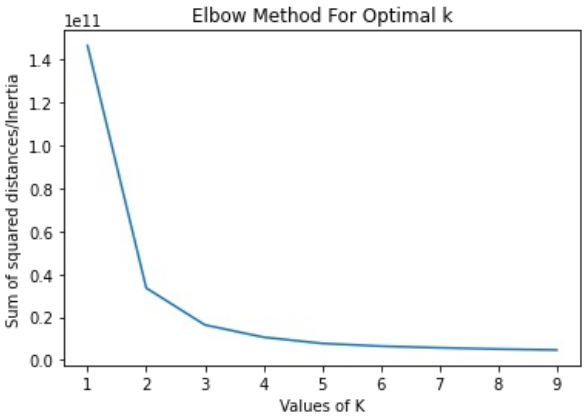
In [289…

```
#Less than 0.5 and close to 0 so the data is sort of uniformly distributed.. and not clustered
```

In [290…

```
Sum_of_squared_distances = []
K = range(1,10)
for num_clusters in K :
    kmeans = KMeans(n_clusters=num_clusters)
    kmeans.fit(data1)
    Sum_of_squared_distances.append(kmeans.inertia_)

plt.plot(K,Sum_of_squared_distances)
plt.xlabel('Values of K')
plt.ylabel('Sum of squared distances/Inertia')
plt.title('Elbow Method For Optimal k')
plt.show()
```



In [291…

```
kmeans = KMeans(n_clusters=3)
kmeans.fit(data1)
print(kmeans.cluster_centers_)
print(kmeans.cluster_centers_.shape)
```

```
[[1.26061823e+03 3.52596045e-01 1.75455383e+02 6.44869251e+00
  2.17310236e+00 2.08620777e+00 2.23481380e+00]
 [2.19183497e+02 3.37939112e-01 1.89201053e+02 6.78211903e+00
  2.20960323e+00 2.16248244e+00 2.24476826e+00]
 [2.42018690e+03 3.28751015e-01 1.78301526e+02 6.27683416e+00
  2.19055202e+00 2.11484627e+00 2.24197924e+00]]
```

(3, 7)

In [292…
```
data1['k-m label']=kmeans.fit_predict(data1)
```

In [293…
```
data1.head()
```

Out[293…

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

In [298…
```
#Aggolomerative clustering

data2 = data1.drop(columns=['k-m label'])
```

In [299…
```
data2.shape
```

Out[299…
(147140, 7)

In [300…
```
z=data2.sample(frac=0.025)
```

In [301…
```
z.shape
```

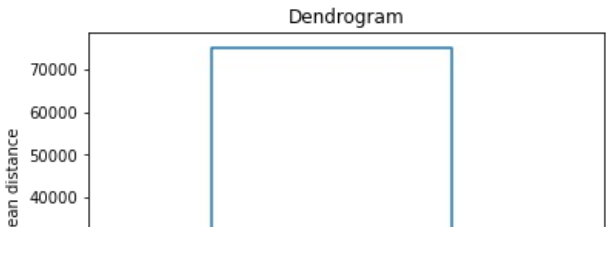
Out[301…
(3678, 7)

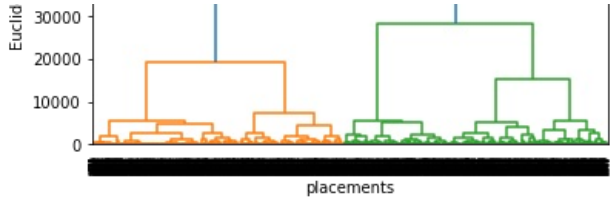
In [302…
```
z.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3678 entries, 41013 to 118341
Data columns (total 7 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   company_hash         3678 non-null   int32
 1   ctc                  3678 non-null   float64
 2   job_position         3678 non-null   int32
 3   years_of_experience  3678 non-null   float64
 4   class                3678 non-null   int64
 5   designation          3678 non-null   int64
 6   tier                 3678 non-null   int64
dtypes: float64(2), int32(2), int64(3)
memory usage: 201.1 KB
```

In [303…
```
import sys
sys.setrecursionlimit(100000)
```

In [304…
```
import scipy.cluster.hierarchy as sch
dendrogrm = sch.dendrogram(sch.linkage(z, method = 'ward'))
plt.title('Dendrogram')
plt.xlabel('placements')
plt.ylabel('Euclidean distance')
plt.show()
```





In [305…
```
from sklearn.cluster import AgglomerativeClustering
model = AgglomerativeClustering(n_clusters=3, affinity='euclidean', linkage='ward')
model.fit(z)
```

Out[305…
```
AgglomerativeClustering(n_clusters=3)
```

In [306…
```
z['Aglo-label'] = model.fit_predict(z)
```

In [307…
```
z.head()
```

Out[307…

	company_hash	ctc	job_position	years_of_experience	class	designation	tier	Aglo-label
41013	1730	0.312312	230	6.0	1	2	2	1
102154	532	0.240240	0	4.0	2	1	2	0
85564	2557	0.510510	105	8.0	1	2	1	2
28380	1039	0.102102	478	1.0	3	2	3	1
74006	49	0.204204	478	8.0	2	2	3	0

In [ ]:
```
```