Problem Statement:

profiling the best companies and job positions to work for from the Scaler database. clustering the segment of learners on the basis of their job profile, company, and other features. Ideally, these clusters should have similar characteristics.

Data Dictionary:

'Unnamed 0' - Index of the dataset

Email_hash - Anonymised Personal Identifiable Information (PII)

Company_hash - This represents an anonymized identifier for the company, which is the current employer of the learner.

orgyear - Employment start date

CTC - Current CTC

Job_position - Job profile in the company

CTC_updated_year - Year in which CTC got updated (Yearly increments, Promotions)

```
from google.colab import files
files.upload()
```

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import math as m
from sklearn.impute import KNNImputer
from sklearn.impute import SimpleImputer
import warnings

warnings.filterwarnings('ignore')
```

```
df1=pd.read_csv('scaler_clustering.csv')
df1.head()
```

	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

```
df1.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      153279 non-null  object 
 6   ctc_updated_year  205843 non-null  float64
dtypes: float64(2), int64(2), object(3)
memory usage: 11.0+ MB
```

```
df1.shape
```

```
(205843, 7)
```

```
#rename the column name
df1.rename(columns={'orgyear':'Employement_Start_Year','ctc_updated_year':'Current_Year','ctc':'CTC','company_hash':'Company'},inplace=True)
```

```
df1.head()
```

	Unnamed: 0	Company	email_hash	Employement_Start_Year	CTC	job_position	Current_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

```
df1['Unnamed: 0'].describe()
```

	Unnamed: 0
count	205843.000000
mean	103273.941786
std	59741.306484
min	0.000000
25%	51518.500000
50%	103151.000000
75%	154992.500000
max	206922.000000

```
#we can drop the column Unnamed: 0,since there is no correlation with driver churn
df1.drop('Unnamed: 0',axis=1,inplace=True)
```

```
len(df1[df1.duplicated()])
```

```
34
```

```
df2 = df1.drop_duplicates()
df2.shape
```

```
(205809, 6)
```

Data Processing

```
df2['email_hash'].duplicated().sum()
```

```
52366
```

```
df2[df2['email_hash'] == 'bbace3cc586400bbc65765bc6a16b77d8913836cfc98b77c05488f02f5714a4b']
```

	Company	email_hash	Employement_Start_Year	CTC	job_position	Current_Year
24109	oxej ntwyzgrgsxto rxbxnta	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	2018.0	720000	NaN	2020.0
45984	oxej ntwyzgrgsxto rxbxnta	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	2018.0	720000	Support Engineer	2020.0
72315	oxej ntwyzgrgsxto rxbxnta	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	2018.0	720000	Other	2020.0
102915	oxej ntwyzgrgsxto rxbxnta	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	2018.0	720000	FullStack Engineer	2020.0
117764	oxej ntwyzgrgsxto	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7	2018.0	720000	Data Analyst	2020.0

```
#keep the first row alone
df2 = df2.groupby('email_hash').first().reset_index()
```

```
#Checking for null values
df2.isna().sum()
```

	0
email_hash	0
Company	32
Employement_Start_Year	78
CTC	0
job_position	20225
Current_Year	0

```
#SimpleImputer(strategy = "most_frequent").fit_transform(df[['company_hash']])
```

Missing Values Treatment

Mean Imputation on Categorical values

```
df2['Company'] = df2['Company'].fillna('Not Available')
```

```
df2.isna().sum()
```

	0
email_hash	0
Company	0
Employement_Start_Year	78
CTC	0
job_position	20225
Current_Year	0

```
#find the company with lenght 1,seems to invalid company
(df2['Company'].str.len()==1).sum()
```

```
14
```

```
#remove the rows with company length is 1
df = df2[~((df2['Company'].str.len())==1)]
```

```
df.shape
```

```
(153429, 6)
```

Imputaion in Job position

Label encoding

```
df['job_position'] = df['job_position'].fillna('na')
enc_nom = (df.groupby('job_position').size()) / len(df)*10000
```

```
df['job_position'+'_encode'] = df['job_position'].apply(lambda x : enc_nom[x])


enc_nom1 = (df.groupby('Company').size()) / len(df)*10000
df['Company'+'_encode'] = df['Company'].apply(lambda x : enc_nom1[x])


df.head()
```

	email_hash	Company	Employement_Start_Year	CTC	job_position	Current_Year	job_posit
0	00003288036a44374976948c327f246fdbdf0778546904...	bxwqgogen	2012.0	3500000	Backend Engineer	2019.0	2
1	0000aaa0e6b61f7636af1954b43d294484cd151c9b3cf6...	nqsn axsxnvr	2013.0	250000	Backend Engineer	2020.0	2
2	0000d58fbc18012bf6fa2605a7b0357d126ee69bc41032...	gunhb	2021.0	1300000	FullStack Engineer	2019.0	1
3	000120d0c8aa304fcf12ab4b85e21feb80a342cfea03d4...	bxwqgotbx wgqugqvnxgz	2004.0	2000000	FullStack Engineer	2021.0	1
4	00014d71a389170e668ba96ae8c1f9d991591acc899025...	fvrbvgn nymo	2009.0	3400000	na	2018.0	1

```
df.isna().sum()
```

	0
email_hash	0
Company	0
Employement_Start_Year	78
CTC	0
job_position	0
Current_Year	0
job_position_encode	0
Company_encode	0

Outlier Treatment

```
sorted(df['Employement_Start_Year'].unique().astype(int))
```

```
1973,
1976,
1977,
1979,
1981,
1982,
1984,
1985,
1986,
1987,
```

```
 2008,
 2009,
 2010,
 2011,
 2012,
 2013,
 2014,
 2015,
 2016,
 2017,
 2018,
 2019,
 2020,
 2021,
 2022,
 2023,
 2024,
 2025,
 2026,
 2027,
 2028,
 2029,
 2031,
 2101,
 2106,
 2107,
 2204,
 20165]
```

```
#remove the learner whose employement start year is before 1970 and 2024 since they are future date and very old past dates,So removing outlier
df=df.loc[(df['Employement_Start_Year']  > 1980) & (df['Employement_Start_Year'] <= 2024)]
```

```
df.isna().sum()
```

	0
email_hash	0
Company	0
Employement_Start_Year	0
CTC	0
job_position	0
Current_Year	0
job_position_encode	0
Company_encode	0

Create new Features

```
#create a column year of experince
df['YOE']=df['Current_Year']-df['Employement_Start_Year']
```

```
sorted(df['YOE'].unique().astype(int))
```

```
[-8,
 -6,
 -5,
 -4,
 -3,
 -2,
 -1,
 0,
 1,
 2,
 3,
 4,
 5,
 6,
 7,
 8,
 9,
 10,
```

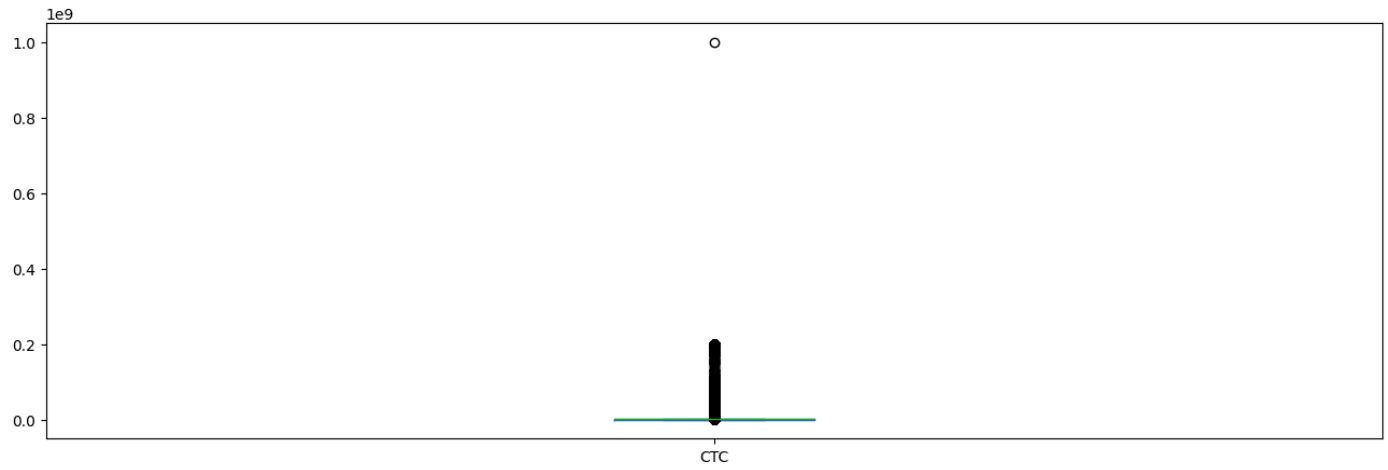
```
        11,
        12,
        13,
        14,
        15,
        16,
        17,
        18,
        19,
        20,
        21,
        22,
        23,
        24,
        25,
        26,
        27,
        28,
        29,
        30,
        31,
        32,
        33,
        34,
        35,
        36,
        38,
        39]
```

```
# we can remove learner whose experince is less that 0,So removing outlier
df=df.loc[(df['YOE']  >= 0)]
```

```
#Categorize the income
df['CTC'].describe()
```

	CTC
count	1.461200e+05
mean	2.457657e+06
std	1.278655e+07
min	2.000000e+00
25%	5.500000e+05
50%	9.699990e+05
75%	1.700000e+06
max	1.000150e+09

```
#seems to be CTC have outliers
df['CTC'].plot.box(figsize=(16,5))
plt.show()
```



We can see outlier are seens to need to remove it

```
dfdd=df.copy()
```

```
df = df[df['CTC'] > 702475]
```

```
print(df.shape)
```

```
(92505, 9)
```

```
#Treating outlier using IQR method
Q1=df['CTC'].quantile(0.25)
Q3=df['CTC'].quantile(0.75)
IQR=Q3-Q1
```

```
#remove outlier from df
df=df[~((df['CTC']<(Q1-1.5*IQR)) | (df['CTC']>(Q3+1.5*IQR)))]
```

```
df.shape
```

```
(86417, 9)
```

```
df.head()
```

	email_hash	Company	Employement_Start_Year	CTC	job_position	Current_Year	job_posit
0	00003288036a44374976948c327f246fdbdf0778546904...	bxwqgogen	2012.0	3500000	Backend Engineer	2019.0	2
3	000120d0c8aa304fcf12ab4b85e21feb80a342cfea03d4...	bxwqgotbx wgqugqvnxgz	2004.0	2000000	FullStack Engineer	2021.0	1
4	00014d71a389170e668ba96ae8e1f9d991591acc899025...	fvrbvqn rvmo	2009.0	3400000	na	2018.0	1
6	00022dc29c7f77032275182b883d4f273ea1007aefc437...	xzeqvwrgha ntwyzgrgsxto	2016.0	750000	Frontend Engineer	2019.0	
7	00036c2c5212d88d07acdc5bda7eef5653f8b09bbe30b7...	ocu xnivz gbvz	2011.0	2300000	Other	2021.0	1

Next steps: Generate code with df | View recommended plots | New interactive sheet

Create a CTC columns which tells whether a learner got any CTC update on current year

```
def func_flag(x):
  if x==2021:
    return 'Yes'
  else:
    return 'No'

df['CTC_UPDATE_FLAG']=df['Current_Year'].apply(lambda x:func_flag(x))


#cretae a ctc update or not column for the current year
df['CTC_UPDATE_FLAG'].value_counts()
```

CTC_UPDATE_FLAG	count
No	65645
Yes	20772

```
bins = [702475, 1500000, 2000000, 4250000]
group = ['Low', 'Average','High']


#catecorize the learner based on the CTC they getting currently
df["income_bin"] = pd.cut(df['CTC'], bins, labels=group)


df["income_bin"].value_counts()
```

income_bin	count
Low	49775
High	21194
Average	15448

Exploratory Data Analysis

Univariate Analysis

```
df.head()
```

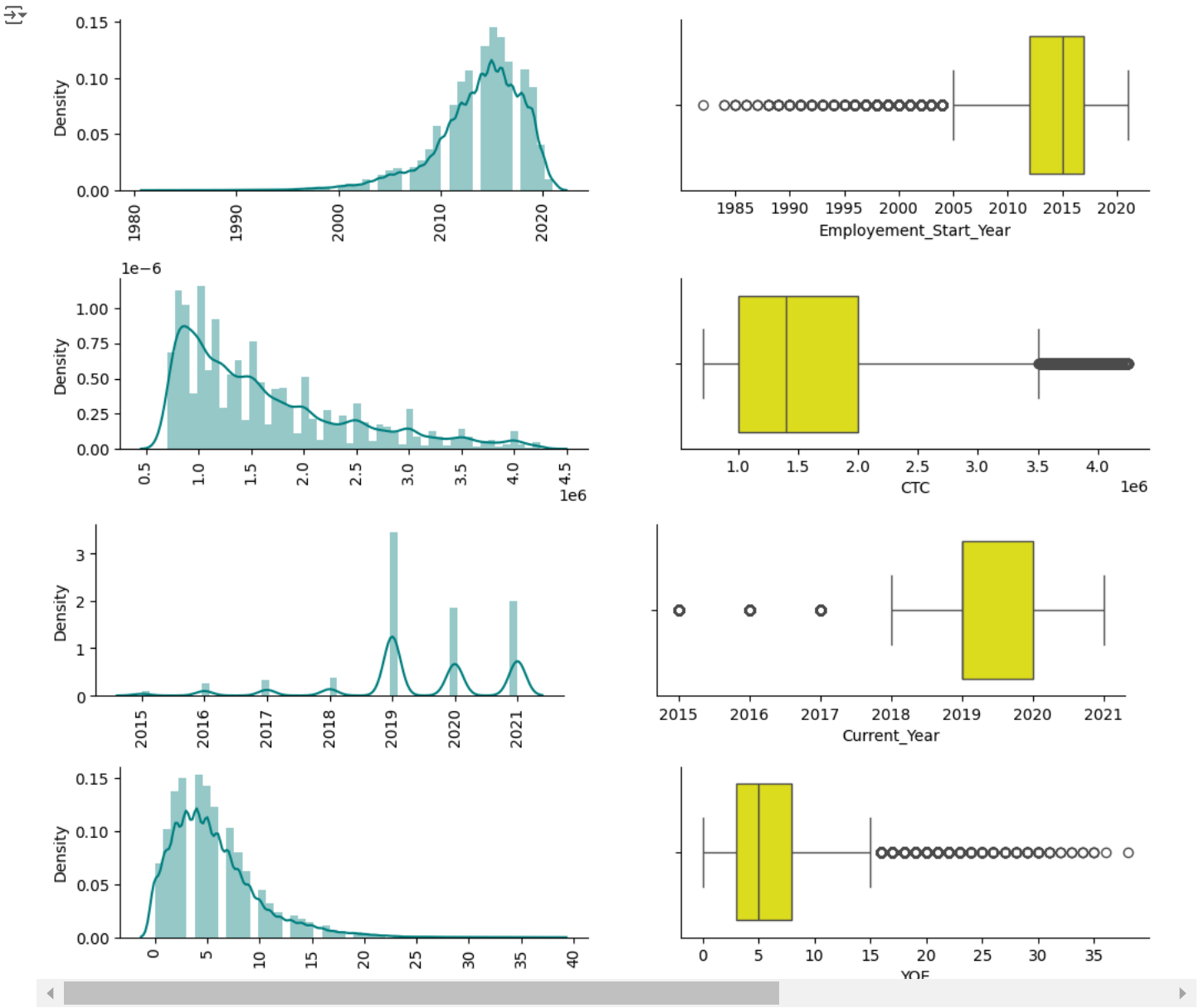
	email_hash	Company	Employement_Start_Year	CTC	job_position	Current_Year	job_posit
0	00003288036a44374976948c327f246fdbdf0778546904...	bxwqgogen	2012.0	3500000	Backend Engineer	2019.0	2
3	000120d0c8aa304fcf12ab4b85e21feb80a342cfea03d4...	bxwqgotbx wgqugqvnxgz	2004.0	2000000	FullStack Engineer	2021.0	1
4	00014d71a389170e668ba96ae8e1f9d991591acc899025...	fvrbvqn rvmo	2009.0	3400000	na	2018.0	1
6	00022dc29c7f77032275182b883d4f273ea1007aefc437...	xzeqvwrgha ntwyzgrgsxto	2016.0	750000	Frontend Engineer	2019.0	
7	00036c2c5212d88d07acdc5bda7eef5653f8b09bbe30b7...	ocu xnivz gbvz	2011.0	2300000	Other	2021.0	1

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
num_cols=['Employement_Start_Year', 'CTC', 'Current_Year', 'YOE']
num_cols
```

```
['Employement_Start_Year', 'CTC', 'Current_Year', 'YOE']
```

```
for i in num_cols:
    plt.figure(figsize=(12,2))
    plt.subplot(121)
    sns.distplot(x=df[i],kde=True,color='teal')
    plt.title('')
    plt.xticks(rotation=90)

    plt.subplot(122)
    sns.boxplot(x=df[i],color='yellow')
    plt.title('')
    sns.despine()
    plt.show()
```



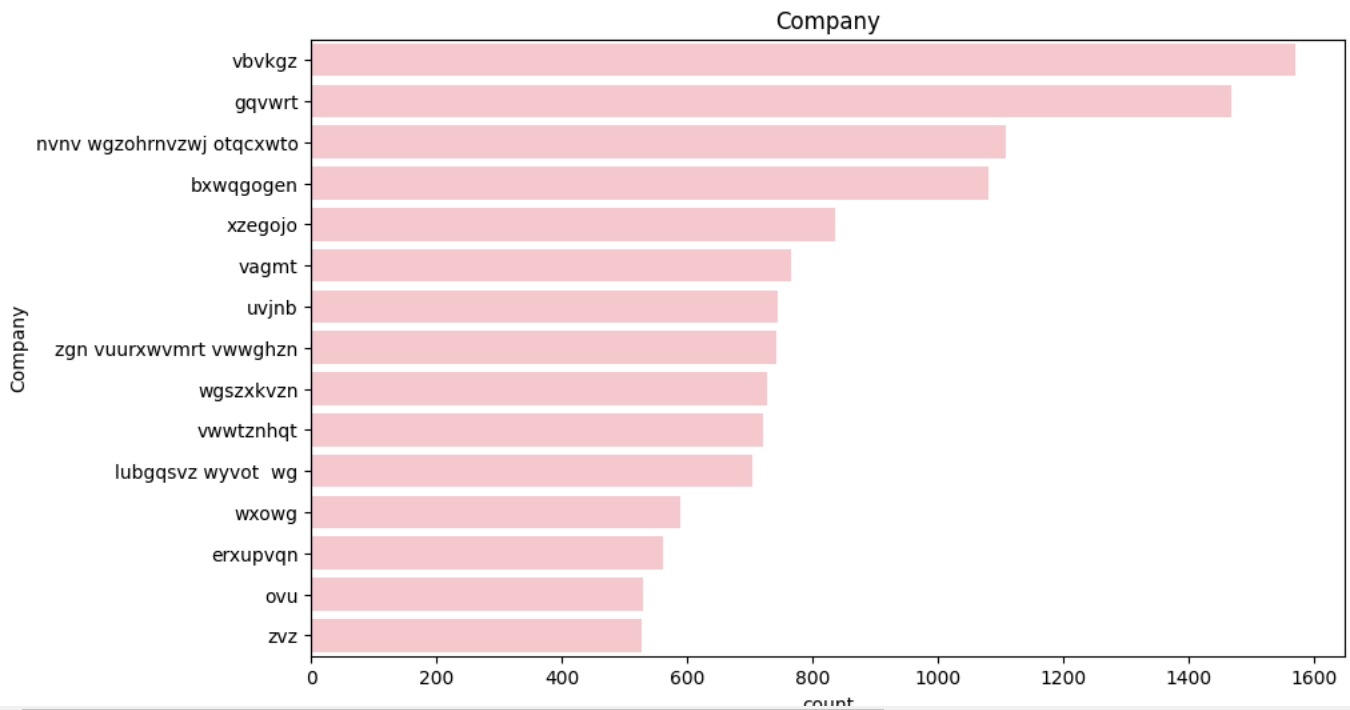
From the above distplot and box plot,we can see that learners start year is fall between 2010 and 2020,so we can see most of the company were started that time.

Most of the learners are getting there latest CTC revised between the year 2019 and 2021.

The majority od the learners experience is between 0 to 13 years.

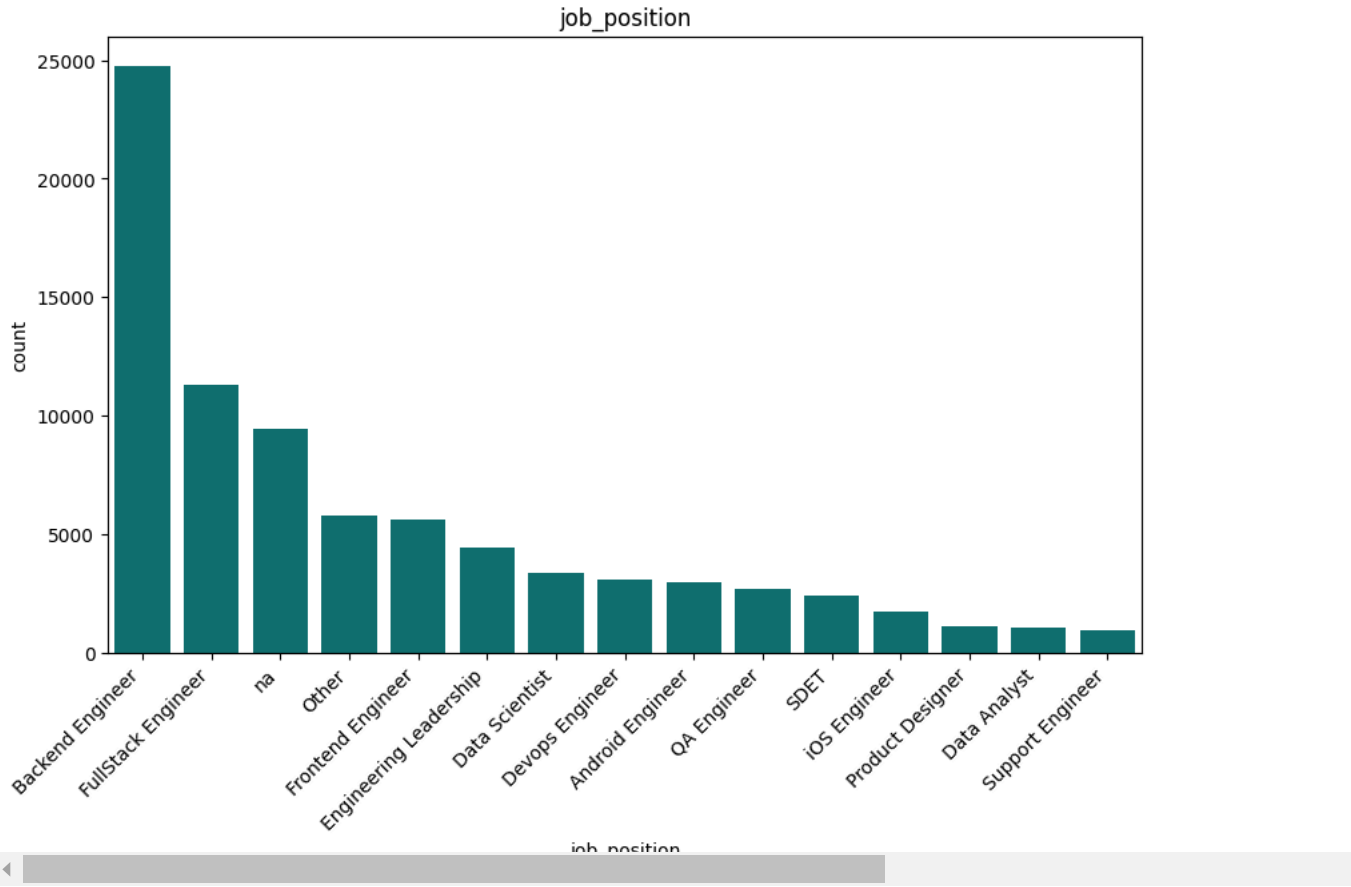
Univariate analysis on Categorial variables

```
#check the count of company_hash
plt.figure(figsize=(10,6))
plt.title('Company')
sns.countplot(y='Company',data=df,order=pd.value_counts(df['Company']).iloc[:15].index,color='pink')
plt.show()
```



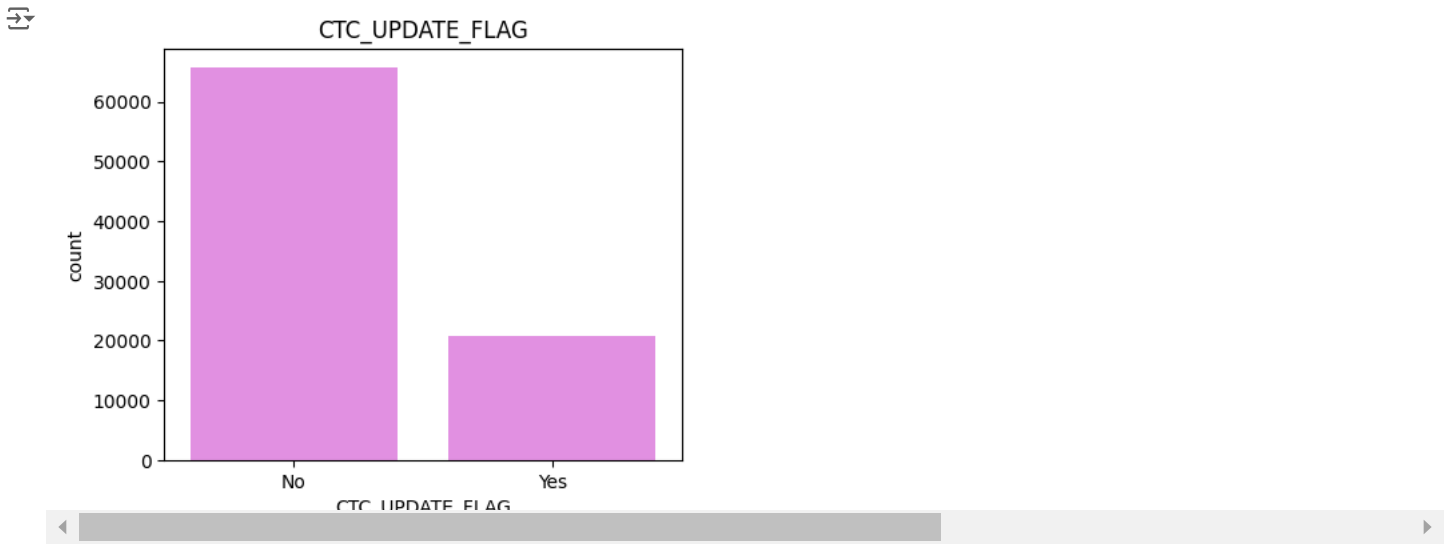
We can see the Top company where most of the learners are from.

```
#check the count of job_position
plt.figure(figsize=(10,6))
plt.title('job_position')
plt.xticks(rotation=45,ha='right')
sns.countplot(x='job_position',data=df,order=pd.value_counts(df['job_position']).iloc[:15].index,color='teal')
plt.show()
```



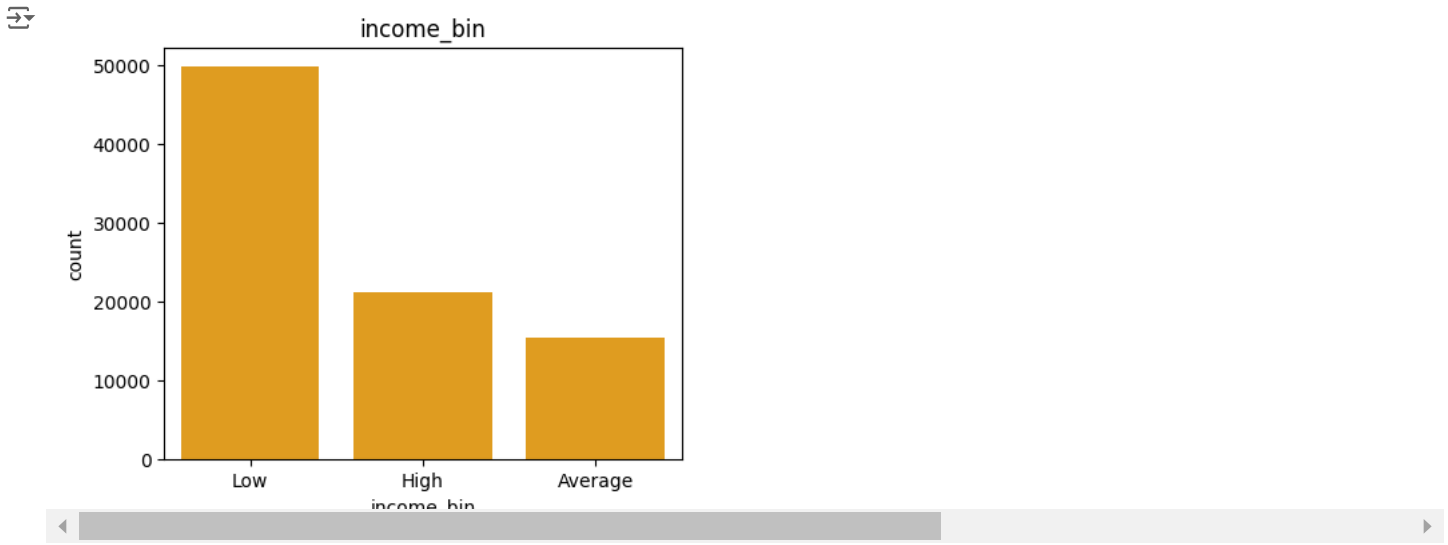
We can see the Top 15 job roles of the learners are **Backend Engineer,Fullstack Enginerr** and also we can see most of the learners are unemployed before joining the scaler acedemy

```
#check the count of CTC_UPDATE_FLAG
plt.figure(figsize=(5,4))
plt.title('CTC_UPDATE_FLAG')
sns.countplot(x='CTC_UPDATE_FLAG',data=df,order=pd.value_counts(df['CTC_UPDATE_FLAG']).iloc[:15].index,color='violet')
plt.show()
```



From the above graph we can see majority of the learner **doesn't get any CTC revise** in the recent years.

```
#check the count of income_bin
plt.figure(figsize=(5,4))
plt.title('income_bin')
sns.countplot(x='income_bin',data=df,order=pd.value_counts(df['income_bin']).iloc[:15].index,color='orange')
plt.show()
```

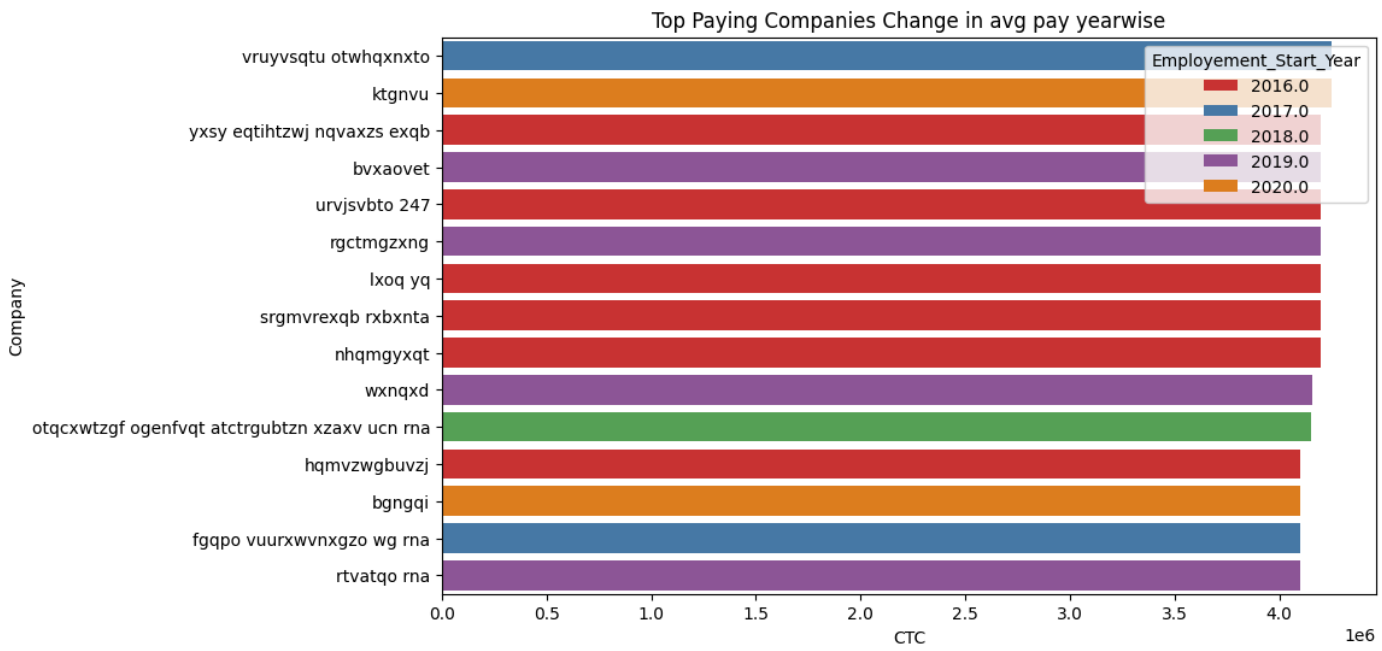


We can see most of the learners are getting **low income** ,hence that might be the reason to join the scaler academy.And can see equal amount of learners are getting low and high salary.

Bivariate Analysis

```
#Top Paying Companies Change in avg pay yearwise
tmp = df[df['Employement_Start_Year'] >= 2016]
tmp_top = tmp.groupby(['Company','Employement_Start_Year'])['CTC'].mean().reset_index().sort_values('CTC',ascending=False)[:15]
plt.figure(figsize=(10,6))
```

```
sns.barplot(data=tmp_top,x='CTC',y='Company',hue='Employement_Start_Year',palette='Set1').set(title="Top Paying Companies Change in avg pay yearwise")
plt.show()
```




```
list(tmp_top['Company'])
```

```
['vruyvsqtu otwhqxnxto',
 'ktgnvu',
 'yxsy eqtihtzwj nqvaxzs exqb',
 'bvxaovet',
 'urvjsvbto 247',
 'rgctmgzxng',
 'lxoq yq',
 'srgmvrexqb rxbxnta',
 'nhqmgyxqt',
 'wxnqxd',
 'otqcxwtzgf ogenfvqt atctrgubtzn xzaxv ucn rna',
 'hqmvzwgbuvzj',
 'bgngqi',
 'fgqpo vuurxwvnxgzo wg rna',
 'rtvatqo rna']
```

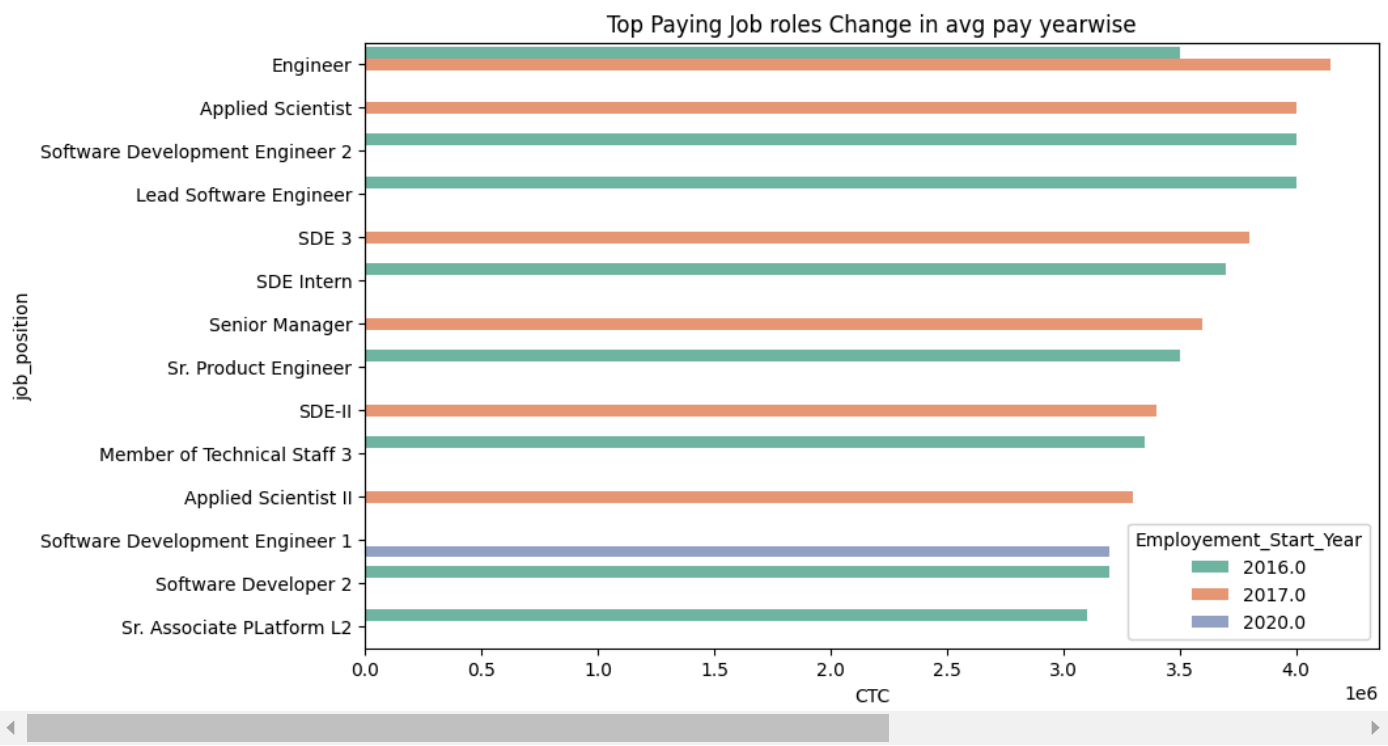
The above are the company which are giveing **high** amount of CTC to the learners

```
tmp_bot = tmp.groupby(['Company'])['CTC'].mean().reset_index().sort_values('CTC',ascending=False)[-15:]
tmp_bot
```

	Company	CTC
2641	nggnyox	710000.0
4685	sqvbgzn ucn rna	710000.0
7293	xzeqvogen ntwyzgrgsxto rna	710000.0
5833	vnntbxn ovqr	710000.0
5100	ttkjrxet xzw	710000.0
240	atruyx ntwyzgrgsxto	710000.0
1170	egqmxzvqj ntwyzgrgsxto ucn rna	710000.0
966	ctqatznhb	710000.0
3429	onvzurho ntwyzgrgsxto	710000.0
2752	nqvzoaxsb	710000.0
1776	h sqg wvuxnvr	710000.0
2753	nqvzoevon xzaxv ucn rna	710000.0
288	avngbo	710000.0
2763	nqvzoutqetwn ogrhnxgzo xzaxv ucn rna	710000.0
2831	ntptzrxsvn ogrhnxgzo ugxeynt rybxnta	705000.0

Next steps: Generate code with tmp_bot | View recommended plots | New interactive sheet

```
#Top Paying Job roles Change in avg pay yearwise
Job_top = tmp.groupby(['job_position','Employement_Start_Year'])['CTC'].mean().reset_index().sort_values('CTC',ascending=False)[:15]
plt.figure(figsize=(10,6))
sns.barplot(data=Job_top,x='CTC',y='job_position',hue='Employement_Start_Year',palette='Set2').set(title="Top Paying Job roles Change in avg pay yearwi
plt.show()
```



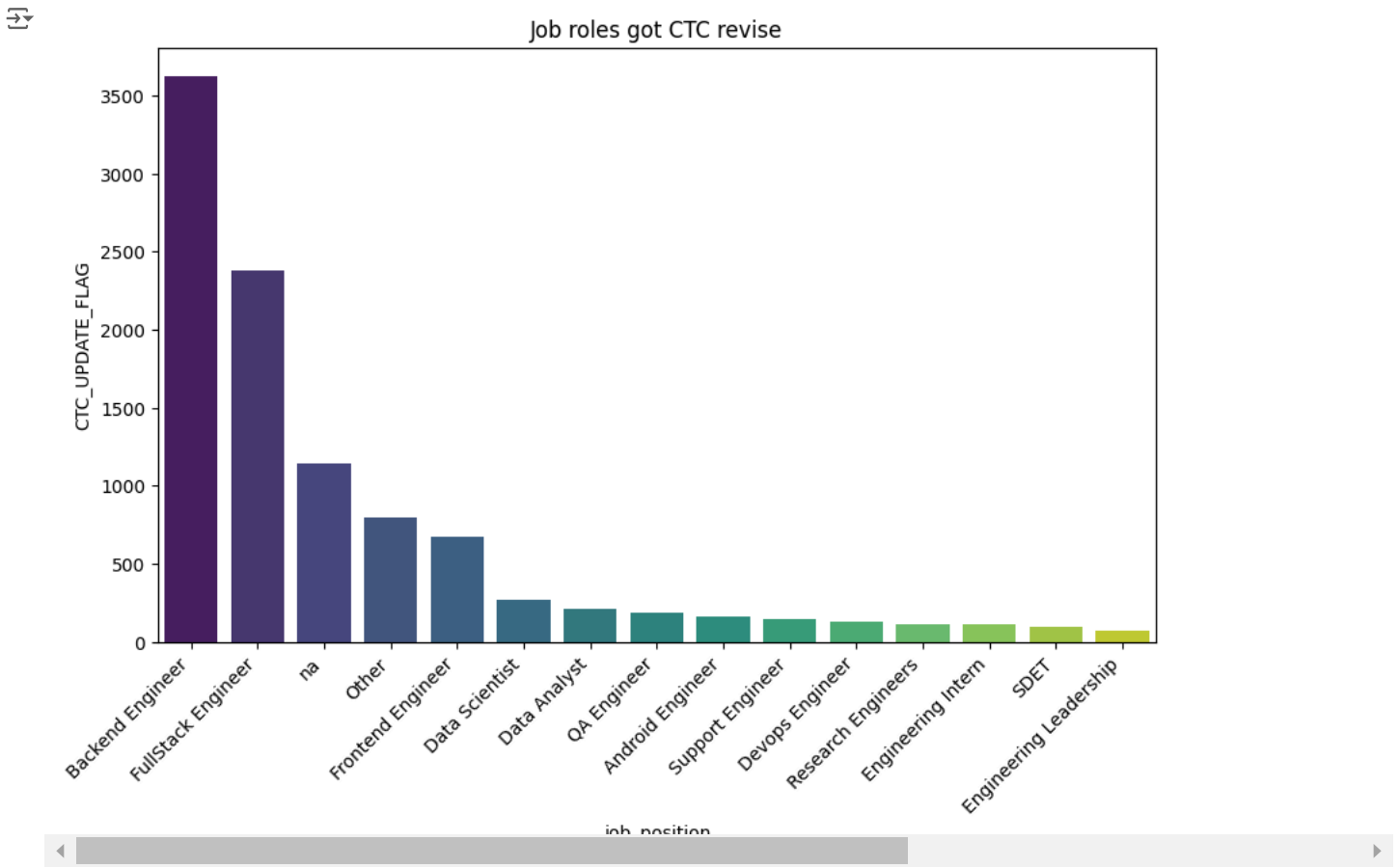

```
list(Job_top['job_position'])
```

```
['Engineer',
 'Applied Scientist',
 'Software Development Engineer 2',
 'Lead Software Engineer',
 'SDE 3',
 'SDE Intern',
 'Senior Manager',
 'Engineer',
 'Sr. Product Engineer',
```

```
 'SDE-II',
 'Member of Technical Staff 3',
 'Applied Scientist II',
 'Software Development Engineer 1',
 'Software Developer 2',
 'Sr. Associate PLatform L2']
```

The above are the job roles which are giveing **high** amount of CTC to the learners

```
tmp1=tmp[tmp['CTC_UPDATE_FLAG']=='Yes']
Job_CTC_rev_top = tmp1.groupby(['job_position',])['CTC_UPDATE_FLAG'].count().reset_index().sort_values('CTC_UPDATE_FLAG',ascending=False)[:15]
plt.figure(figsize=(10,6))
sns.barplot(data=Job_CTC_rev_top,x='job_position',y='CTC_UPDATE_FLAG',palette='viridis').set(title="Job roles got CTC revise")
plt.xticks(rotation=45,ha='right')
plt.show()
```



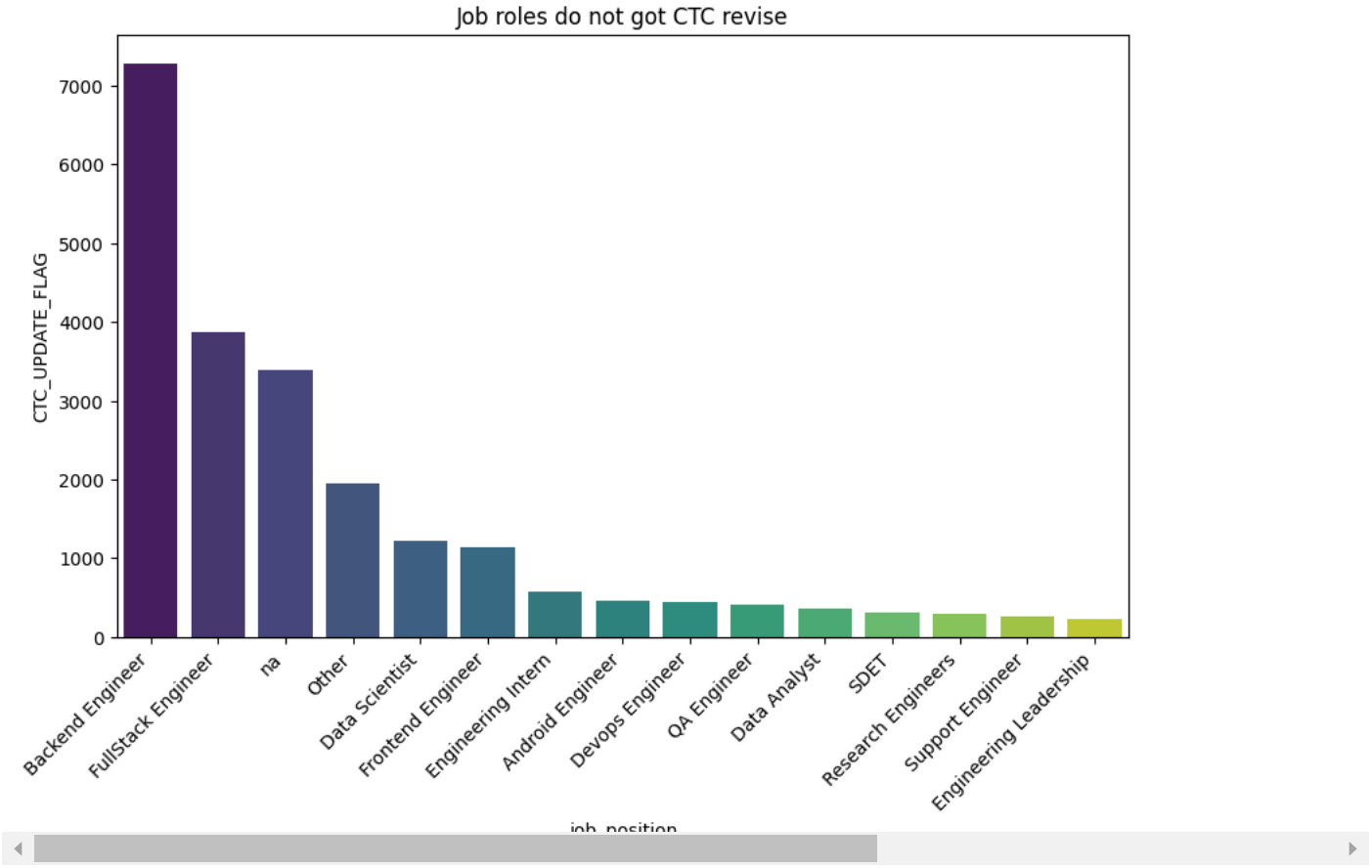

```
list(Job_CTC_rev_top['job_position'])
```

```
['Backend Engineer',
 'FullStack Engineer',
 'na',
 'Other',
 'Frontend Engineer',
 'Data Scientist',
 'Data Analyst',
 'QA Engineer',
 'Android Engineer',
 'Support Engineer',
 'Devops Engineer',
 'Research Engineers',
 'Engineering Intern',
 'SDET',
 'Engineering Leadership']
```

Above are the list job giving CTC revise

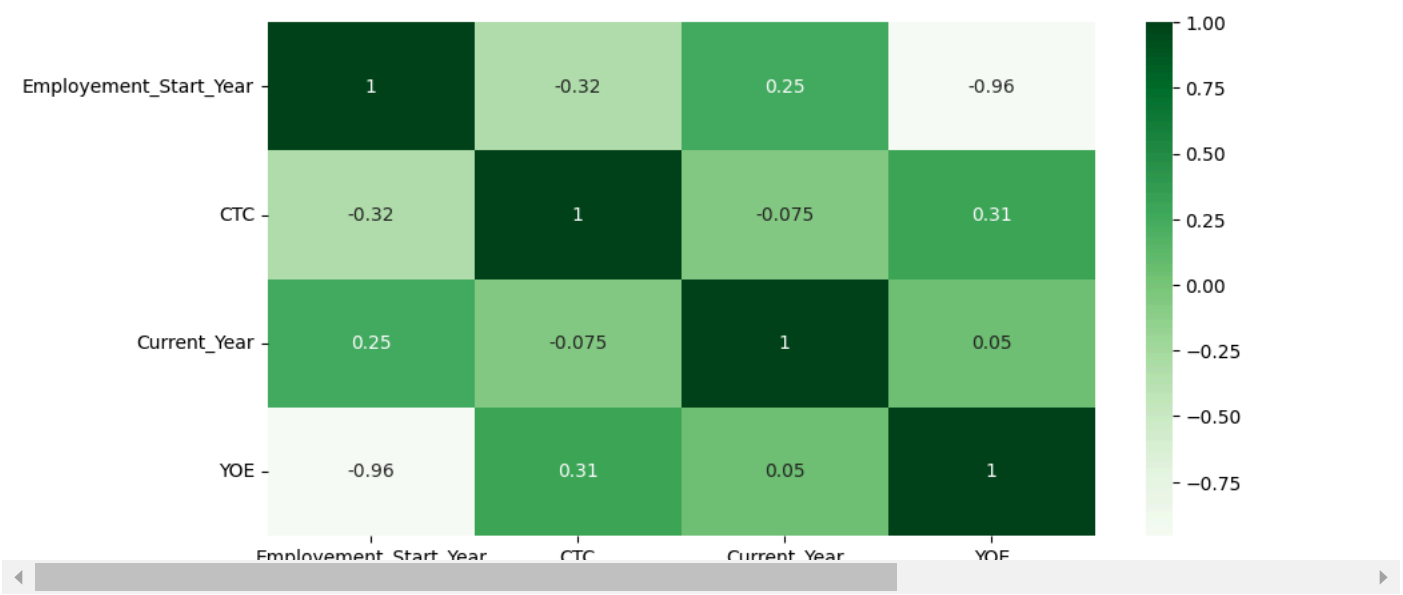
```
tmp2=tmp[tmp['CTC_UPDATE_FLAG']=='No']
Job_CTC_rev_not = tmp2.groupby(['job_position',])['CTC_UPDATE_FLAG'].count().reset_index().sort_values('CTC_UPDATE_FLAG',ascending=False)[:15]
plt.figure(figsize=(10,6))
```

```
sns.barplot(data=Job_CTC_rev_not,x='job_position',y='CTC_UPDATE_FLAG',palette='viridis').set(title="Job roles do not got CTC revise")
plt.xticks(rotation=45,ha='right')
plt.show()
```



Correlation among independent variables

```
corr = df[num_cols].corr()
plt.figure(figsize=(10,5))
sns.heatmap(corr,annot=True,cmap='Greens')
plt.show()
```



We can see The CTC is high for more experince learner and and there is more chance of geting slary hike for those learners

```
dateda=df.copy()
```

```
grp = ['Company','job_position','YOE']
data_tmp1 = dateda.groupby(grp).agg({'CTC':['mean','median','min','max','count']}).reset_index()
data_tmp1.columns  = ["{} {}".format(b_, a_) if a_ not in grp else "{}".format(a_) for a_, b_ in zip(data_tmp1.columns.droplevel(1), data_tmp1.columns.
data_tmp1.head(100).tail(50)

datatmp = dateda.merge(data_tmp1[['Company','job_position','YOE', 'mean CTC']],on=['Company', 'job_position', 'YOE'],how='left')



col1 = 'CTC'
col2 = 'mean CTC'
conditions  = [ datatmp[col1] > datatmp[col2], datatmp[col1] == datatmp[col2], datatmp[col1] < datatmp[col2] ]
choices     = [ 1, 2, 3 ]

datatmp['Designation'] = np.select(conditions, choices, default=np.nan)


grp = ['Company','job_position']
data_tmp1 = datatmp.groupby(grp).agg({'CTC':[('mean2','mean'),'median','min','max','count']}).reset_index()
data_tmp1.columns  = ["{} {}".format(b_, a_) if a_ not in grp else "{}".format(a_) for a_, b_ in zip(data_tmp1.columns.droplevel(1), data_tmp1.columns.
data_tmp1.head(100).tail(50)

datatmp = datatmp.merge(data_tmp1[grp + ['mean2 CTC']],on=grp,how='left')



col1 = 'CTC'
col2 = 'mean2 CTC'
conditions  = [ datatmp[col1] > datatmp[col2], datatmp[col1] == datatmp[col2], datatmp[col1] < datatmp[col2] ]
choices     = [ 1, 2, 3 ]

datatmp['Class'] = np.select(conditions, choices, default=np.nan)


grp = ['Company']
data_tmp1 = datatmp.groupby(grp).agg({'CTC':[('mean3','mean'),'median','min','max','count']}).reset_index()
data_tmp1.columns  = ["{} {}".format(b_, a_) if a_ not in grp else "{}".format(a_) for a_, b_ in zip(data_tmp1.columns.droplevel(1), data_tmp1.columns.
data_tmp1.head(100).tail(50)

datatmp = datatmp.merge(data_tmp1[grp + ['mean3 CTC']],on=grp,how='left')



col1 = 'CTC'
col2 = 'mean3 CTC'
conditions  = [ datatmp[col1] > datatmp[col2], datatmp[col1] == datatmp[col2], datatmp[col1] < datatmp[col2] ]
choices     = [ 1, 2, 3 ]

datatmp['Tier'] = np.select(conditions, choices, default=np.nan)


datatmp['diff_desig'] = datatmp['CTC'] - datatmp['mean CTC']
datatmp['diff_class'] = datatmp['CTC'] - datatmp['mean2 CTC']
datatmp['diff_tier'] = datatmp['CTC'] - datatmp['mean3 CTC']


datatmp.head()
```

	email_hash	Company	Employement_Start_Year	CTC	job_position	Current_Year	job_posit
0	00003288036a44374976948c327f246fdbdf0778546904...	bxwqgogen	2012.0	3500000	Backend Engineer	2019.0	2
1	000120d0c8aa304fcf12ab4b85e21feb80a342cfea03d4...	bxwqgotbx wgqugqvnxgz	2004.0	2000000	FullStack Engineer	2021.0	1
2	00014d71a389170e668ba96ae8e1f9d991591acc899025...	fvrbvqn rvmo	2009.0	3400000	na	2018.0	1
3	00022dc29c7f77032275182b883d4f273ea1007aefc437...	xzeqvwrgha ntwyzgrgsxto	2016.0	750000	Frontend Engineer	2019.0	
4	00036c2c5212d88d07acdc5bda7eef5653f8b09bbe30b7...	ocu xnivz gbvz	2011.0	2300000	Other	2021.0	1

Next steps: Generate code with datatmp | View recommended plots | New interactive sheet

. Which companies dominate in Tier 1 and why might this be the case?

```
df_Tier1 = datatmp[datatmp['Tier']==1].sort_values(by='diff_tier',ascending=False)
df_Tier1.head(5)
```

	email_hash	Company	Employement_Start_Year	CTC	job_position	Current_Year	job_po
76140	e15abfd41c005995728191e49ef001e83e813cd3ed5104...	nvnv wgzohrnvzwj otqcxwto	2015.0	4240000	Support Engineer	2020.0	
59545	b022b84623593cc38a3c1d39d4545b368a7b5f286be1c7...	nvnv wgzohrnvzwj otqcxwto	2015.0	4200000	na	2019.0	
49005	90d5114ca752c55babef2c517ac8b17aaee3d9ff5740de...	nvnv wgzohrnvzwj otqcxwto	2018.0	4200000	Backend Engineer	2020.0	
54733	a1c1c8919e2918b24241a40271e02381daf199c61d7a3b...	wgszxkvzn	2005.0	4200000	Program Manager	2021.0	
70630	d13d7376e9ced16b4e250d0643f9139f8b36a62847f71b...	dvcxtzn xzegqbvnxgz ojontbo	2003.0	4200000	Engineering Leadership	2019.0	

Next steps: Generate code with df_Tier1 View recommended plots New interactive sheet

nvnv wgzohrnvzwj otqcxwto,wgszxkvzn is the company dominate in Tier 1 becuase they are the high paying company

Top 10 employees (earning more than most of the employees in the company) - Tier 1

```
datatmp[datatmp['Tier'] == 1].sort_values('diff_tier',ascending=False).head(10)[['email_hash','CTC','mean3 CTC']]
```

	email_hash	CTC	mean3 CTC
76140	e15abfd41c005995728191e49ef001e83e813cd3ed5104...	4240000	1.051181e+06
59545	b022b84623593cc38a3c1d39d4545b368a7b5f286be1c7...	4200000	1.051181e+06
49005	90d5114ca752c55babef2c517ac8b17aaee3d9ff5740de...	4200000	1.051181e+06
54733	a1c1c8919e2918b24241a40271e02381daf199c61d7a3b...	4200000	1.144022e+06
70630	d13d7376e9ced16b4e250d0643f9139f8b36a62847f71b...	4200000	1.147773e+06
31636	5d872e52cb535a71fc75a5a97e779bb4c1554d0baa920d...	4200000	1.155724e+06
14802	2b10e1d996c6ab5e175eea35ca25ea7afbaacd1237ab64...	4200000	1.155724e+06
47702	8d0ed00904247626f5557f5983feeb5a0567d7726eea39...	4200000	1.177756e+06
31813	5dff6a65d548553262b6a289f014b2b72a5d47ff6dfa5c...	4170000	1.165011e+06
45603	86b90dd64ddb663ea35be98422947a01ba9ab837fb76df...	4000000	1.051181e+06

Top 10 employees of data science in each company earning more than their peers - Class 1

```
datatmp[(datatmp['Class'] == 1) & (datatmp['job_position'] == 'Data Scientist')].sort_values('diff_class',ascending=False).head(10)[['job_position','em
```

	job_position	email_hash	CTC	mean2 CTC
81246	Data Scientist	f04a0228e5af6f8f6ecc33e089892e80d85b3c749b3244...	4000000	1.533750e+06
56202	Data Scientist	a63f3f44de7586430615a8a9bd13d41e7b0d541ca0f690...	4200000	1.862000e+06
16839	Data Scientist	31616edfc502824631b11793313d35d5bb2288319dcb25...	3800000	1.513842e+06
21432	Data Scientist	3efbb8c4d67b4a4c6ba4c639cd84e9ff98b85d5f57d82f...	3979999	1.716000e+06
33497	Data Scientist	62f705ba342cb9e51117446a5522c2e42c14db27b9b20e...	4250000	2.025000e+06
83352	Data Scientist	f67ae342b7431f7ab05eca998d904647b02711538aa839...	3750000	1.565556e+06
83480	Data Scientist	f6e8c41a40ec308c996d498e22729359d2b564cae037a0...	3500000	1.410000e+06
191	Data Scientist	009ded427ebcb5c2fb1970017a683693a7abef0fa96f5e...	3900000	1.834333e+06
79488	Data Scientist	eb35a5d34977c6135372e46d6cc4f85332f1a4f9578bd5...	4080000	2.020000e+06
36067	Data Scientist	6aa8cfeb5b98da66158e0af4ca8869362174abdba84a02...	3200000	1.233235e+06

Bottom 10 employees of data science in each company earning less than their peers - Class 3

```
datatmp[(datatmp['Class'] == 3) & (datatmp['job_position'] == 'Data Scientist')].sort_values('diff_class',ascending=True).head(10)[['job_position','ema
```

	job_position	email_hash	CTC	mean2 CTC
46472	Data Scientist	8969b3943b8e5d77bcca59a1f206078eb2ea64a42f61d0...	850000	3.250000e+06
38279	Data Scientist	7146830df2009a0fdf8a93626e53a9870e33b729f44884...	1200000	3.007143e+06
29827	Data Scientist	5847bf4aaa694dda8dac8c6b3f18cb990403619fca0df3...	750000	2.475000e+06
40057	Data Scientist	767bcba126b7b5ea20715fc280a8df74cd19e7843e4091...	1200000	2.824444e+06
35857	Data Scientist	6a0061b2d6741e96e69db990dfffcf08700b1fa38d208c...	969999	2.590000e+06
37807	Data Scientist	6fe2c6b27c366842ae182ff52e945a2a36cc696727c6fd...	960000	2.580000e+06
48379	Data Scientist	8f00376fe6d2c78b903fcdf95b4294452fdaff2adbd65e...	930000	2.465000e+06
66359	Data Scientist	c4bc19d7af570ba40bfe228d77cbd69f8afe97e58b4dad...	900000	2.385714e+06
73240	Data Scientist	d8d55f0943c1d59773d2395dd4af985c6164e1faa34bf8...	750000	2.187500e+06
14508	Data Scientist	2a3136f6e2d03a3dbfa3f683e4ae1b744b4815a8e0177c...	1700000	3.125000e+06

Bottom 10 employees (earning less than most of the employees in the company)- Tier 3

```
datatmp[(datatmp['Tier'] == 3)].sort_values('diff_tier',ascending=True).head(10)[['email_hash','CTC','mean3 CTC']]
```

	email_hash	CTC	mean3 CTC
12117	2322345290a1926df62347d45f06b68932e219cb010bf8...	850000	3.262923e+06
64034	bda6e0f742115289a27f304078935331a5563d90c91461...	750000	2.929000e+06
15902	2e7e946b56a245338d8da1daf60ef851031c9964cffd25...	950000	2.950000e+06
4333	0c535bb44414d62cab133425339bd7e156ec79823899ae...	810000	2.770000e+06
49879	935480e039d80833292d858a553a4bc0f628b9b97ce9ec...	900000	2.840543e+06
73438	d96a6540ff59456abe30f51f68e954388b1f6922c4bb0c...	900000	2.840543e+06
19337	38d71a484d7663f7c14df8432620bbbab718933173a295...	1368000	3.262923e+06
70255	d034e386dbce817ee1ea099b161379d3341af0a16573d8...	800000	2.683125e+06
35987	6a6d1a4452505b678e264700fd0c28f247c4522d27f112...	770000	2.637273e+06
2611	077fd3f95d8dbf89c112a8eca6601db3729f51b53b57e0...	720000	2.577054e+06

```
datatmp[(datatmp['YOE'].isin([5,6,7]))&datatmp['Company'].isin(['sgrabvz ovwyo'])].sort_values('diff_desig',ascending=True).head(10)[['email_hash','CTC
```

	email_hash	CTC	mean3 CTC
19117	3838c8da0c3bfd4de1218a0a09badde9868a86e496f571...	1200000	2.577054e+06
46068	883371bd326d4288c50a2fee87d4c66b502cbcaa86e934...	1200000	2.577054e+06
9248	1ac1e1a99bf62a2ecbcb5830dfc63df53eb4aa43f675b5...	1400000	2.577054e+06
9262	1aca31126bf62d09c550a156035cf989c3bbc5a2ebf1cd...	1200000	2.577054e+06
15262	2c732532049292f28ad9eb8d010f5a13c4eb57ede33843...	1400000	2.577054e+06
44373	832d90f0a8a1076adeb4493bf8867cd9a843facc22732a...	1500000	2.577054e+06
82270	f35723eae980dc41ada45312d8823fbfa8bb5724161d9e...	1700000	2.577054e+06
4670	0d47e3dad72a17107832d2ad22c670aefa31de3d985ff1...	1700000	2.577054e+06
23363	450d25a586750c6ba330b7c383ae773ffca5eaf76f794c...	1800000	2.577054e+06
24042	471bc1235f0f5a2b80260a44eca96766f9db805771d54b...	2500000	2.577054e+06

Top 10 companies (based on their CTC)

```
tmp_top
```

	Company	Employement_Start_Year	CTC
9192	vruyvsqtu otwhqxnxto	2017.0	4250000.0
3208	ktgnvu	2020.0	4250000.0
11762	yxsy eqtihtzwj nqvaxzs exqb	2016.0	4200000.0
1148	bvxaovet	2019.0	4200000.0
8207	urvjsvbto 247	2016.0	4200000.0
6664	rgctmgzxng	2019.0	4200000.0
3487	lxoq yq	2016.0	4200000.0
7205	srgmvrexqb rxbxnta	2016.0	4200000.0
4139	nhqmgyxqt	2016.0	4200000.0
10416	wxnqxd	2019.0	4160000.0
5368	otqcxwtzgf ogenfvqt atctrgubtzn xzaxv ucn rna	2018.0	4150000.0
2752	hqmvzwgbuvzj	2016.0	4100000.0
679	bgngqi	2020.0	4100000.0
2136	fgqpo vuurxwvnxgzo wg rna	2017.0	4100000.0
6817	rbvatgo rna	2019.0	4100000.0

Next steps: Generate code with tmp_top | View recommended plots | New interactive sheet

Top 2 positions in every company (based on their CTC)

```
Job_top = datatmp.groupby(['Company','job_position'])['CTC'].max().reset_index().sort_values(['Company','CTC'],ascending=False)
tmp1 = Job_top.groupby('Company').head(2)[['Company','job_position']][:20]
tmp1
```

	Company	job_position
35705	zzzbzb	Other
35704	zz	Other
35703	zyvzwt wgzohrnxzs tzsxzttqo	Frontend Engineer
35702	zyvzwt fgga qtztfvr eqvzwyxogq yi	na
35701	zyuw rxbxnta	Frontend Engineer
35700	zxzvzxjv sqghu	Engineering Leadership
35699	zxztrtvuo ntwyzgrgsj ogrhnxgzo	Backend Engineer
35690	zxztrtvuo	Backend Architect
35693	zxztrtvuo	Devops Engineer
35688	zxztonvqo xzegqbvnxgz ntwyzgrgsxto ucn rna	na
35687	zxztonvqo xzegqbvnxgz ntwyzgrgsxto rna	Product Designer
35682	zxzlvwvqn	Backend Engineer
35685	zxzlvwvqn	FullStack Engineer
35681	zxzlvpvqn	na
35680	zxzlvbtzngqo	Engineering Leadership
35679	zxzlv cvz	na
35678	zxyxrtzn rxbxnta egqbtqrj muv ntwyzgrgsxto ucn...	SDET
35677	zxyxrtzn rxbxnta	FullStack Engineer
35674	zxyxrtzn ntwyzgrgsxto	Fullstack Engineer
35673	zxyxrtzn ntwyzgrgsxto	Backend Engineer

Next steps: Generate code with tmp1 | View recommended plots | New interactive sheet

Data Modeling

```
df.head()
```

	email_hash	Company	Employement_Start_Year	CTC	job_position	Current_Year	job_posit
0	00003288036a44374976948c327f246fdbdf0778546904...	bxwqgogen	2012.0	3500000	Backend Engineer	2019.0	2
3	000120d0c8aa304fcf12ab4b85e21feb80a342cfea03d4...	bxwqgotbx wgqugqvnxgz	2004.0	2000000	FullStack Engineer	2021.0	1
4	00014d71a389170e668ba96ae8e1f9d991591acc899025...	fvrbvqn rvmo	2009.0	3400000	na	2018.0	1
6	00022dc29c7f77032275182b883d4f273ea1007aefc437...	xzeqvwrgha ntwyzgrgsxto	2016.0	750000	Frontend Engineer	2019.0	
7	00036c2c5212d88d07acdc5bda7eef5653f8b09bbe30b7...	ocu xnivz gbvz	2011.0	2300000	Other	2021.0	1

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
#Since CTC is having high range of values we will transform the values using log tranformation for Data modeling
df['CTC_log'] = np.log(df['CTC'])


#convert CTC_UPDATE_FLAG to bool
df['CTC_UPDATE_FLAG'] = df['CTC_UPDATE_FLAG'].map({'Yes':True,'No':False})


#convert income_bin to int
df['income_bin_int'] = df['income_bin'].map({'Low':1,'Average':2,'High':3})



df['income_bin_int'].value_counts()
```

income_bin_int	count
1	49775
3	21194
2	15448

```
df['income_bin_int']=df['income_bin_int'].astype(int)


df.columns
```

```
Index(['email_hash', 'Company', 'Employement_Start_Year', 'CTC',
       'job_position', 'Current_Year', 'job_position_encode', 'Company_encode',
       'YOE', 'CTC_UPDATE_FLAG', 'income_bin', 'CTC_log', 'income_bin_int'],
      dtype='object')
```

```
df_model=df.copy()


#drop few columns
df_model.drop(['email_hash','job_position','Company','CTC','income_bin'],axis=1,inplace=True)


df_model.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 86417 entries, 0 to 153442
Data columns (total 8 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Employement_Start_Year  86417 non-null  float64
 1   Current_Year           86417 non-null  float64
 2   job_position_encode    86417 non-null  float64
 3   Company_encode         86417 non-null  float64
 4   YOE                    86417 non-null  float64
```

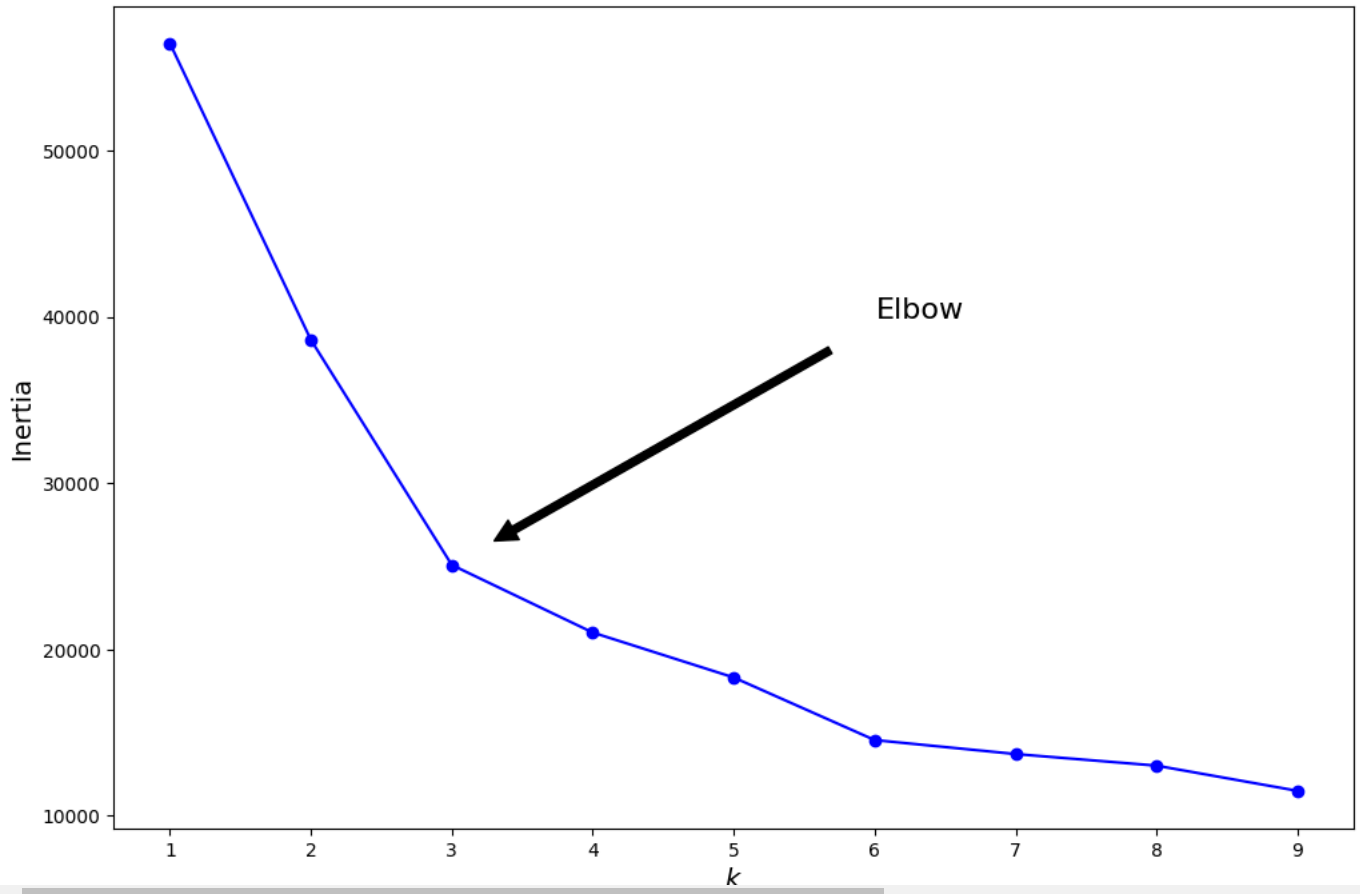
```
 5   CTC_UPDATE_FLAG         86417 non-null  bool
 6   CTC_log                 86417 non-null  float64
 7   income_bin_int          86417 non-null  int64
dtypes: bool(1), float64(6), int64(1)
memory usage: 5.4 MB
```

```
from sklearn.impute import KNNImputer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.cluster import MiniBatchKMeans, KMeans
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
```

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
scaler.fit(df_model)
df_model_sc=scaler.transform(df_model)
```

```
#fidn K using elbow method
kmeans_per_k = [KMeans(n_clusters=k, random_state=42).fit(df_model_sc)
                for k in range(1, 10)]

inertias = [model.inertia_ for model in kmeans_per_k]
```

```
#fidn K using elbow method
plt.figure(figsize=(12, 8))
plt.plot(range(1, 10), inertias, "bo-")
plt.xlabel("$k$", fontsize=14)
plt.ylabel("Inertia", fontsize=14)
plt.annotate('Elbow',
             xy=(3, inertias[2]),
             xytext=(0.55, 0.55),
             textcoords='figure fraction',
             fontsize=16,
             arrowprops=dict(facecolor='black', shrink=0.1)
            )
plt.show()
```



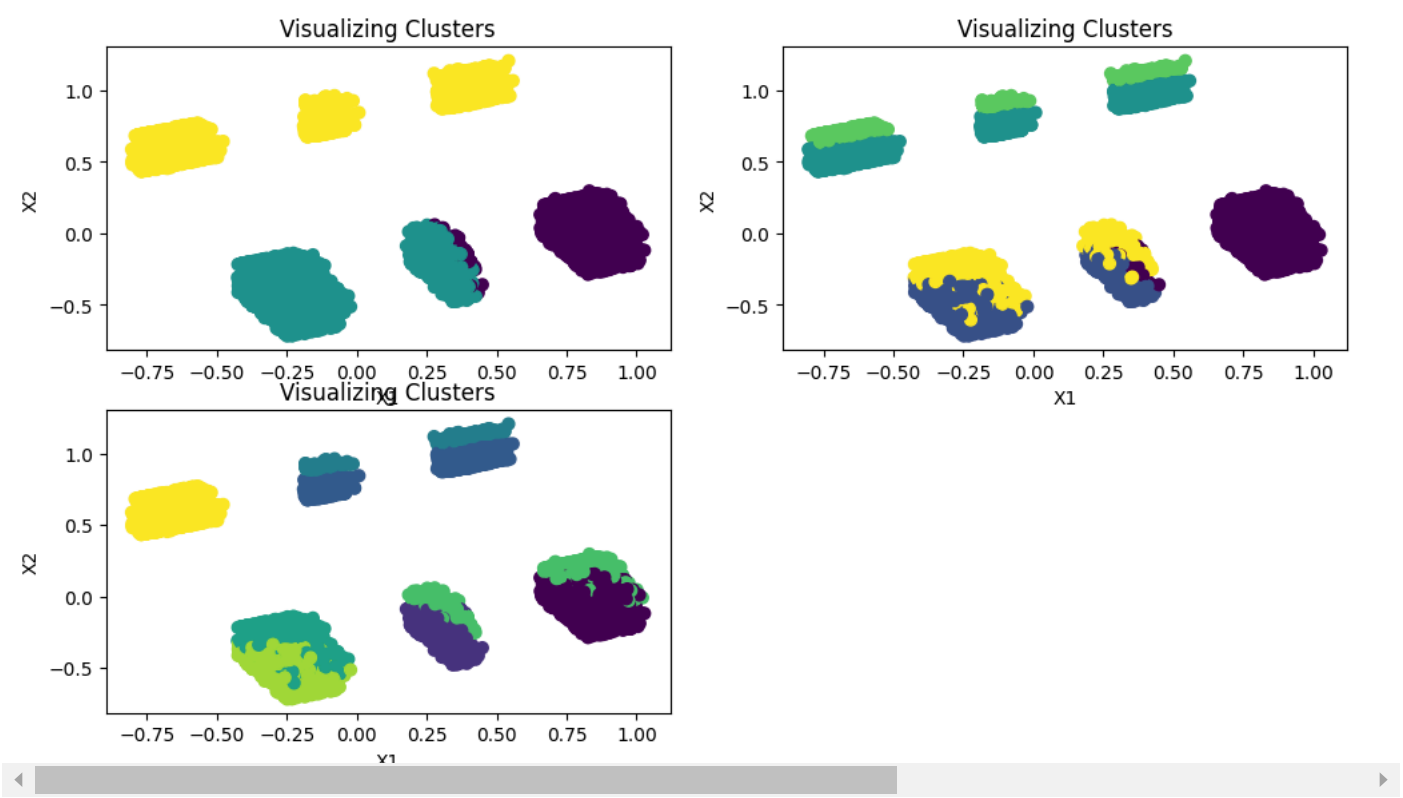
K=3 is best clustering

```
#find K using PCA for visulization
from sklearn.decomposition import PCA

pca = PCA(2)

components_pca = pca.fit_transform(df_model_sc)


def viz_clusters(clusters):
    plt.scatter(clusters['X1'], clusters['X2'], c=clusters['label'], s = 40)
    plt.xlabel('X1')
    plt.ylabel('X2')
    plt.title('Visualizing Clusters')


kmeans_iter1 = KMeans(n_clusters=3, init="random", n_init=1,
                      algorithm="lloyd", random_state=0)
kmeans_iter2 = KMeans(n_clusters=5, init="random", n_init=1,
                      algorithm="lloyd", random_state=0)
kmeans_iter3 = KMeans(n_clusters=8, init="random", n_init=1,
                      algorithm="lloyd", random_state=0)
kmeans_iter1.fit(df_model_sc)
kmeans_iter2.fit(df_model_sc)
kmeans_iter3.fit(df_model_sc)
```

```
KMeans
KMeans(init='random', n_init=1, random_state=0)
```

```
clusters_1 = pd.DataFrame(components_pca, columns=['X1', 'X2'])
clusters_1['label'] = kmeans_iter1.labels_


clusters_2 = pd.DataFrame(components_pca, columns=['X1', 'X2'])
clusters_2['label'] = kmeans_iter2.labels_
```

```
clusters_3 = pd.DataFrame(components_pca, columns=['X1', 'X2'])
clusters_3['label'] = kmeans_iter3.labels_


plt.figure(figsize=(12,10))

plt.subplot(321)
viz_clusters(clusters_1)

plt.subplot(322)
viz_clusters(clusters_2)

plt.subplot(323)
viz_clusters(clusters_3)
```




```
#print inertia_ for all k
print('inertia_ for k=3',kmeans_iter1.inertia_)
print('inertia_ for k=5',kmeans_iter2.inertia_)
print('inertia_ for k=8',kmeans_iter3.inertia_)
```

```
inertia_ for k=3 25886.295889154506
inertia_ for k=5 19069.112453758575
inertia_ for k=8 12183.266132079181
```

For decrease in the cluster the BCSS is increase,mens both are inversly proportional.K=3 seems good

We can see cluster size with 3 give better clusing that n_cluster =5 or 8

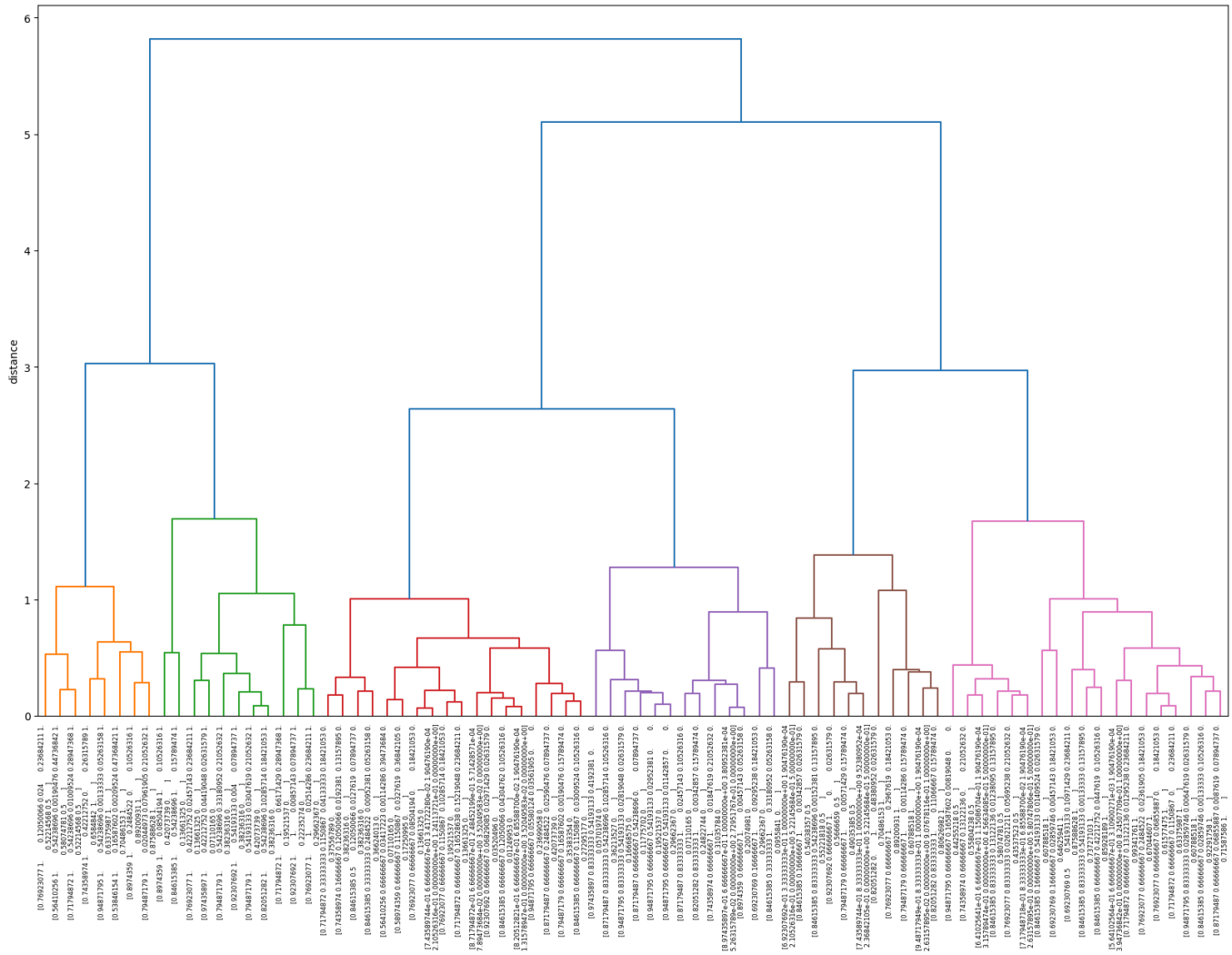
Hierarchical CLustering

```
sample_80=df_model_sc[:80]


# import hierarchical clustering libraries
import scipy.cluster.hierarchy as sch

#Refer https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html#scipy.cluster.hierarchy.linkage
Z = sch.linkage(sample_80, method='ward') #linkage = ward


len(sample_80)
```

```
80
```

```
fig, ax = plt.subplots(figsize=(20, 12))
sch.dendrogram(Z, labels=sample_80, ax=ax, color_threshold=2)
plt.xticks(rotation=90)
ax.set_ylabel('distance')
```

```
Text(0, 0.5, 'distance')
```



Insights

From the above distplot and box plot,we can see that learners start year is fall between 2010 and 2020,so we can see most of the company were started that time.

Most of the learners are getting there latest CTC revised between the year 2019 and 2021.

The majority od the learners experience is between 0 to 13 years.

majority of the learner doesn't get any CTC revise in the recent years.

learners are getting low income ,hence that might be the reason to join the scaler academy.And can see equal amount of learners are getting low and high salary.

'vruyvsqtu otwhqxnxto', 'ktgnvu', 'yxsy eqtihtzwj nqvaxzs exqb', 'bvxaovet', 'urvjsvbto 247 are the hightes paying companies.

'Engineer', 'Applied Scientist', 'Software Development Engineer 2', 'Lead Software Engineer', 'SDE 3' are the hightes paying job roles.

Backend Engineer', 'FullStack Engineer','Frontend Engineer', 'Data Scientist' are top hike based job roles in recent years.

We can see The CTC is high for more experince learner and and there is more chance of geting slary hike for those learners.

vnv wgzohrnvzwj otqcxwto,wgszxkvzn is the company dominate in Tier 1 becuase they are the high paying company.

For decrease in the cluster the BCSS is increase,mens both are inversly proportional.K=3 seems good.

K=3 is best clustering.

Recommendations:

Freshers who want to work on technical side should look for roles related to Backend Engineer, SDET, QA engineer, Dataa Scientist.

Learner can search for new job in these companies 'vruyvsqtu otwhqxnxto', 'ktgnvu', 'yxsy eqtihtzwj nqvaxzs exqb', 'bvxaovet', 'urvjsvbto 247.