Problem Statement

Scaler is an online tech-versity offering intensive computer science & Data Science courses through live classes delivered by tech leaders and subject matter experts. The meticulously structured program enhances the skills of software professionals by offering a modern curriculum with exposure to the latest technologies. It is a product by InterviewBit.

You are working as a data scientist with the analytics vertical of Scaler, focused on profiling the best companies and job positions to work for from the Scaler database. You are provided with the information for a segment of learners and tasked to cluster them on the basis of their job profile, company, and other features. Ideally, these clusters should have similar characteristics.

In [68]:
```
import re
import numpy as np
import pandas as pd
```

In [69]:
```
df = pd.read_csv('Scaler.csv')
df.head(2)
```

Out[69]:

	Unnamed: 0	company_hash	email_hash	orgyear	
0	0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	1
1	1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018.0	

In [70]:
```
pd.DataFrame(df.job_position.value_counts())[50:60]
```

Out[70]:

	job_position
Technology Analyst	10
SDE 3	10
Senior Consultant	9
SDE-1	9
Software Engineer (Full stack)	9
Software Engineer I	8
Senior Engineer	8
Senior Systems Engineer	8
Associate software engineer	8
SDE2	8

Data Preprocessing - Cleaning of all variables

In [71]:
```
df.drop(columns=['email_hash','Unnamed: 0'],inplace=True)
```

In [72]:
```
def preprocess_string(string):
    new_string= re.sub('[^A-Za-z ]+', '', string).lower().strip()
    return new_string

mystring='\tAirtel\\\\&&**() X Labs'
preprocess_string(mystring)
```

Out[72]:
```
'airtel x labs'
```

In [73]:
```
df.job_position.nunique()
```

Out[73]:
```
1017
```

In [74]:
```
df.job_position=df.job_position.apply(lambda x: preprocess_string(str(x)))
df.job_position.nunique()
```

Out[74]:
```
857
```

In [ ]:
```
df.job_position
```

In [75]:
```
df.shape
```

Out[75]:
```
(205843, 5)
```

In [76]:
```
df.drop_duplicates(inplace=True)
df.shape
```

Out[76]:
```
(188247, 5)
```

In [77]:
```
df['company_hash'].value_counts().sort_index()
```

Out[77]:
```
0                                3
01 ojztqsj                       2
05mz exzytvrny uqxcvnt rxbxnta   2
1                                2
1 axsxnvro                       1
                                ..
zyvzwt wgzohrnxzs tzsxzttqo      1
zz                               2
zzb ztdnstz vacxogqj ucn rna     2
zzgato                           1
zzzbzb                           1
Name: company_hash, Length: 37298, dtype: int64
```

In [78]:
```
df.company_hash.nunique()
```

Out[78]:
```
37298
```

In [79]:
```
df.company_hash=df.company_hash.apply(lambda x: preprocess_string(str(x)))
df.company_hash.nunique()
```

Out[79]:
```
37208
```

In [80]:
```
df['company_hash'].value_counts().sort_index()
```

Out[80]:
```
                                                  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 [81]:
```
#removing rows where company or job_position is not available
df=df[ ~((df['company_hash']=='') | (df['job_position']==''))]
```

Filling Null values using Mean Target Inputation for Orgyear

In [82]:
```
df['orgyear'].isnull().sum()
```

Out[82]: 86

In [83]:
```
df['orgyear'].fillna(df.groupby('company_hash')['orgyear'].transform('media
```

In [84]:
```
df = df.loc[~df['orgyear'].isna()]
```

Checking for outliers in orgyear

In [85]:
```
df.orgyear.head()
```

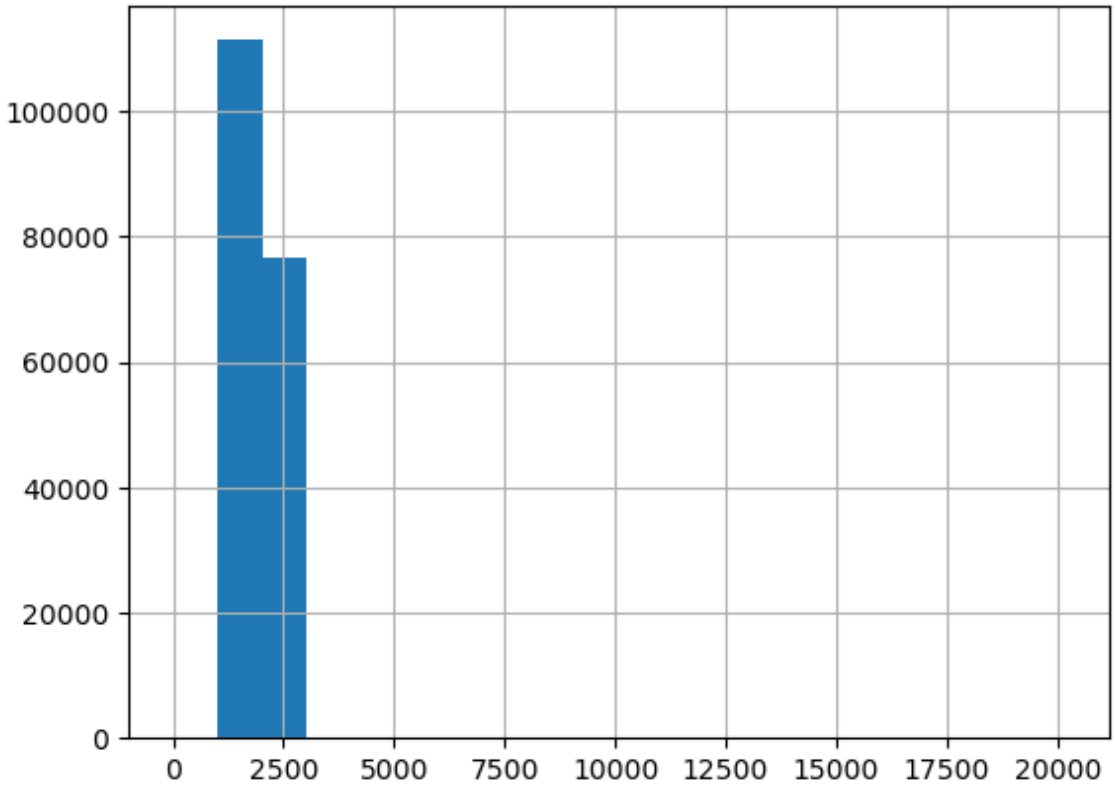
Out[85]:
```
0    2016.0
1    2018.0
2    2015.0
3    2017.0
4    2017.0
Name: orgyear, dtype: float64
```

In [86]:
```
df.orgyear.describe()
```

Out[86]:
```
count    188127.000000
mean       2014.614019
std          66.472744
min           0.000000
25%        2013.000000
50%        2016.000000
75%        2018.000000
max       20165.000000
Name: orgyear, dtype: float64
```

In [87]:
```
#simple understanding
df.orgyear.hist(bins=20)
```

Out[87]:
```
<Axes: >
```



In [88]:
```
df['orgyear'] = df['orgyear'].clip(lower=df.orgyear.quantile(0.01), upper=d
df['ctc'] = df['ctc'].clip(lower=df.ctc.quantile(0.01), upper=df.ctc.quanti
```

In [89]:
```
#We see some 'nan's in job_position
df.loc[df['job_position']=='nan', 'job_position']=np.nan
```

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

In [90]:
```
#making the code
df.company_hash.value_counts()
```

Out[90]:
```
nvnv wgzohrnvzwj otqcxwto     4284
xzegojo                       3043
vbvkgz                        3005
wgszxkvzn                     2262
zgn vuurxwvmrt vwwghzn        2208
                              ...
nojo wvqa ntwy                   1
wtzwgoha ov                      1
sgzxjg                           1
ozvu gz mhoxztoo ogrhnxgzo       1
bvptbjnqxu td vbvkgz             1
Name: company_hash, Length: 37180, dtype: int64
```

In [91]:
```
df.loc[df.groupby('company_hash')['ctc'].transform('count') < 5, 'company_h
```

Creating Years of Experience Columns

In [92]:
```
df['orgyear'].describe()
```

Out[92]:
```
count    188127.000000
mean       2014.891379
std           4.138710
min        2000.000000
25%        2013.000000
50%        2016.000000
75%        2018.000000
max        2021.000000
Name: orgyear, dtype: float64
```

In [93]:
```
df['years_of_experience']=2022-df['orgyear']
```

In [94]:
```
df.drop_duplicates(inplace=True)
df.shape
```

Out[94]:
```
(166852, 6)
```

In [95]:
```
df=df[~df['years_of_experience'].isnull()]
```

In [96]:
```
# update cant be before joining
df['ctc_updated_year'] = df[['ctc_updated_year', 'orgyear']].max(axis=1)
```

In [97]:
```
#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 [98]:
```
df.isnull().sum()
#All good now
```

Out[98]:
```
company_hash           0
orgyear                0
ctc                    0
job_position           0
ctc_updated_year       0
years_of_experience    0
dtype: int64
```

In [99]:
```
df.drop_duplicates(inplace=True)
df.shape
```

Out[99]:
```
(165722, 6)
```

In [100]:
```
df.describe()
```

Out[100]:

	orgyear	ctc	ctc_updated_year	years_of_experience
count	165722.000000	1.657220e+05	165722.000000	165722.000000
mean	2014.784552	1.539193e+06	2019.583755	7.215448
std	4.187307	1.836941e+06	1.325783	4.187307
min	2000.000000	3.000000e+04	2015.000000	1.000000
25%	2013.000000	6.000000e+05	2019.000000	4.000000
50%	2016.000000	1.040000e+06	2020.000000	6.000000
75%	2018.000000	1.800000e+06	2021.000000	9.000000
max	2021.000000	1.500000e+07	2021.000000	22.000000

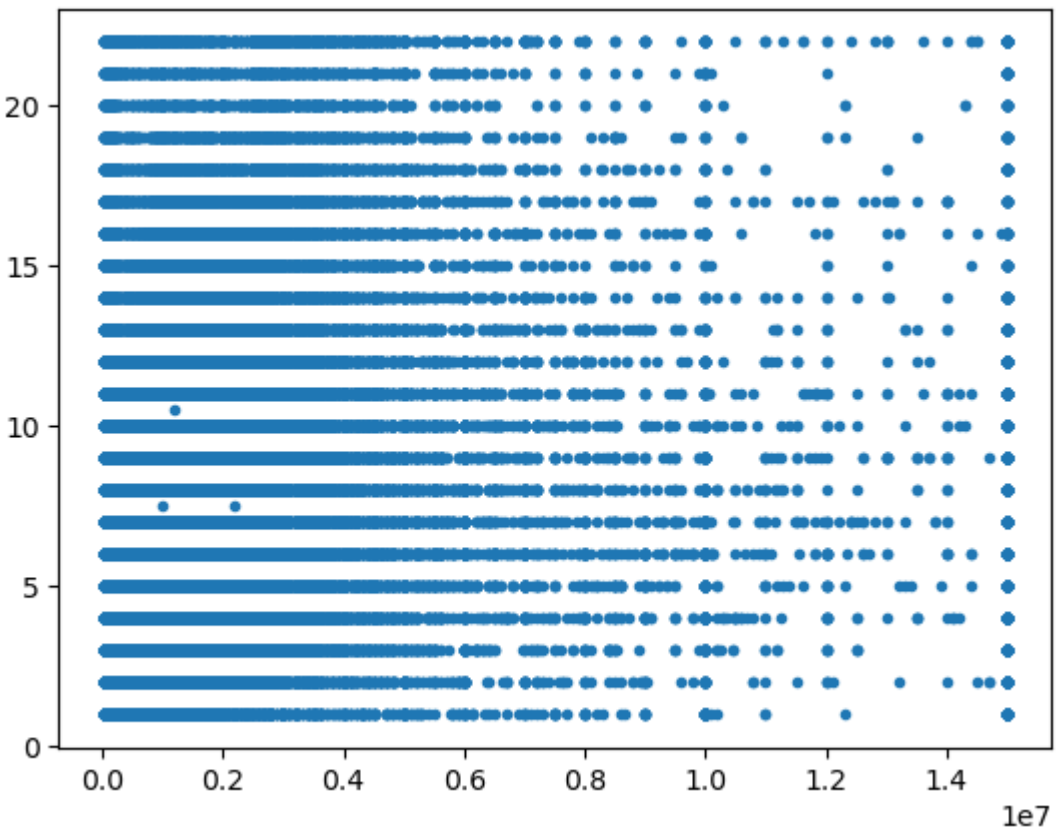
In [101]:
```
df.head(2)
```

Out[101]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience
0	atrgxnnt xzaxv	2016.0	1100000	other	2020.0	6.0
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	fullstack engineer	2019.0	4.0

In [102]:
```
import matplotlib.pyplot as plt
plt.plot(df['ctc'], df['years_of_experience'], '.')
```

Out[102]:
```
[<matplotlib.lines.Line2D at 0x160aeff4970>]
```



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

In [ ]:
```
grouped_c_j_y = df.groupby(['years_of_experience','job_position','company_h
grouped_c_j_y
```

In [ ]:
```
df_cjy=df.merge(grouped_c_j_y, on=['years_of_experience','job_position','co
df_cjy
```

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

In [ ]:
```
def segment(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 [ ]:
```
df_cjy['designation'] =df_cjy.apply(lambda x: segment(x['ctc'],x['50%'],x['
df_cjy.head(2)
```

In [ ]:
```
df_cjy.designation.value_counts(normalize=True)
```

Manual Clustering based on company and job position

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

In [ ]:
```
df_cj=df.merge(grouped_c_j, on=['job_position','company_hash'], how='left')
df_cj.head(2)
```

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

In [ ]:
```
df_cj['classs'] = df_cj.apply(lambda x: segment(x['ctc'],x['50%'],x['75%'])
df_cj.head(2)
```

In [ ]:
```
df_cj.classs.value_counts(normalize=True)
```

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

In [ ]:
```
df_cj.drop(columns=['count','mean','std','min','25%','50%','75%','max'],inp
df_cjy.drop(columns=['count','mean','std','min','25%','50%','75%','max'],in
```

In [ ]:
```
df_cjy_cj=df_cj.merge(df_cjy, on=['company_hash','orgyear','ctc','job_posit
df_cjy_cj
```

Manual Clustering based on comapny

In [ ]:
```
grouped_c = df.groupby(['company_hash'])['ctc'].describe()
df_c = df.merge(grouped_c, on=['company_hash'], how='left')
df_c.head(2)
```

Creating Tier basis on the salary in the companies

In [ ]:
```
df_c['tier'] =df_c.apply(lambda x: segment(x['ctc'],x['50%'],x['75%']),axis
df_c.head(2)
```

In [ ]:
```
df_c.tier.value_counts(normalize=True)
```

In [ ]:
```
df_cjy_cj_c=df_cjy_cj.merge(df_c, on=['company_hash','orgyear','ctc','job_p
df_cjy_cj_c.head(10)
```

In [ ]:
```
df_cjy_cj_c.drop(columns=['count','mean','std','min','25%','50%','75%','max
df_cjy_cj_c.head(2)
```

In [ ]:
```
df_cjy_cj_c.columns
```

In [ ]:
```
X = df_cjy_cj_c[['ctc', 'years_of_experience', 'classs', 'designation', 'ti
```

In [ ]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(X)
X_sc = pd.DataFrame(scaler.transform(X), columns=X.columns, index=X.index)
```

In [ ]:
```
X_sc.shape
```

In [ ]:
```
import scipy.cluster.hierarchy as sch
import matplotlib.pyplot as plt

sample = X_sc.sample(500)
Z = sch.linkage(sample, method='ward')

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

In [ ]:
```
from sklearn.cluster import KMeans

k = 3
kmeans = KMeans(n_clusters=k, random_state=42)
y_pred = kmeans.fit_predict(X_sc)

##coordinates of the cluster centers
# kmeans.cluster_centers_
clusters = pd.DataFrame(X_sc, columns=X.columns)
clusters['label'] = kmeans.labels_
```

In [ ]:
```
X_sc.columns
```

In [ ]:
```
x_axis = 'years_of_experience'
y_axis = 'classs'

plt.scatter(clusters[x_axis], clusters[y_axis], c=clusters['label'], )
plt.scatter(kmeans.cluster_centers_[:, 1], kmeans.cluster_centers_[:, 2], c
plt.xlabel(x_axis)
plt.ylabel(y_axis);
```

In [ ]:
```
import plotly.express as px
fig = px.scatter_3d(clusters, x='years_of_experience', y='ctc', z='tier', c
fig.show()
```

In [ ]:
```
df.columns
```

In [ ]:
```
plt.plot(df['ctc'], df['years_of_experience'], '.')
```

In [ ]:
```
df['ctc'].hist(bins=30)
```

In [ ]:
```
print('Ran')
```

In [ ]:
```

```