In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import image as mpimg
import seaborn as sns
pd.options.display.max_columns = None
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.decomposition import PCA
from sklearn.cluster import MiniBatchKMeans, DBSCAN
from scipy.cluster.hierarchy import linkage, dendrogram
from sklearn.pipeline import make_pipeline
import plotly.express as px
from pyclustertend.hopkins import hopkins
```

```
C:\Users\rauna\Anaconda3\envs\scaler\lib\site-packages\numpy\_distributor_init.py:
30: UserWarning: loaded more than 1 DLL from .libs:
C:\Users\rauna\Anaconda3\envs\scaler\lib\site-packages\numpy\.libs\libopenblas.FB5
AE2TYXYH2IJRDKGDGQ3XBKLKTF43H.gfortran-win_amd64.dll
C:\Users\rauna\Anaconda3\envs\scaler\lib\site-packages\numpy\.libs\libopenblas.GK7
GX5KEQ4F6UYO3P26ULGBQYHGQO7J4.gfortran-win_amd64.dll
  warnings.warn("loaded more than 1 DLL from .libs:"
```

Problem Statement :

Cluster incoming students into different groups based on their employment details like CTC, designation, year of joining etc.

Insights :

- Backend/Fullstack/Frontend Engineers are the most common job positions
- nvnv wgzohrnvzwj otqcxwto is the most common company ~ 4%
- Engineering Leadership is the highest paid job ~ 26Lakh median salary
- bxwqgogen is the company which has highest median salary ~ 26Lakh
- There is very linear relationship between number of years spent at an organization vs their salary
- Median salary for employees who joined in
  - 2000 -> 26 Lakh
  - 2021 -> 6 Lakh
- We got a Hopkins score ~ 0 meaning there is strong clustering tendency in the data
- Kmeans, DBSCAN and Hierarichal clustering techniques all showed presence of 2 clusters in our dataset
- There is a 80-20 split between these two clusters
- Minority class are mostly backend/fullstack engineers or have unidentified (other/nan/misc) job positions

- The minority cluster are the employees who work at the top companies with lower experience and lower ctc
- The majority cluster are the employees who dont work at the top companies but have higher experience and ctc

Recommendations :

- Scaler can create two pitches for these 2 clusters :
  - The majority cluster needs to be shown how scaler can help them get into the top companies of their fields
  - The minority cluster needs to be shown how scaler can help them get into higher positions in their existing companies.
- Scaler can also promote interactions between these two clusters so that they can impart each other with their own learnings
  - Minority cluster can share their experience of working in a big company
  - Majority cluster can share their experience and skills required to grow up in an organization

EDA

In [2]:
```
df = pd.read_csv('clustering.csv')
```

In [3]:
```
df.head()
```

Out[3]:

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

In [4]:
```
df = df.drop(columns=['Unnamed: 0','email_hash'])
```

In [5]:
```
df.shape
```

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

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205843 entries, 0 to 205842
Data columns (total 5 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   company_hash      205799 non-null  object 
 1   orgyear           205757 non-null  float64
 2   ctc               205843 non-null  int64  
 3   job_position      153281 non-null  object 
 4   ctc_updated_year  205843 non-null  float64
dtypes: float64(2), int64(1), object(2)
memory usage: 7.9+ MB
```

In [7]:
```
round(df.isna().mean() * 100,2)
```

Out[7]:
```
company_hash         0.02
orgyear              0.04
ctc                  0.00
job_position        25.53
ctc_updated_year     0.00
dtype: float64
```

- job position has 25% null values
- orgyear and company_hash also has few null values

In [8]:
```
df.describe()
```

Out[8]:

	orgyear	ctc	ctc_updated_year
count	205757.000000	2.058430e+05	205843.000000
mean	2014.882750	2.271685e+06	2019.628231
std	63.571115	1.180091e+07	1.325104
min	0.000000	2.000000e+00	2015.000000
25%	2013.000000	5.300000e+05	2019.000000
50%	2016.000000	9.500000e+05	2020.000000
75%	2018.000000	1.700000e+06	2021.000000
max	20165.000000	1.000150e+09	2021.000000

In [9]:
```
df.describe(include='O')
```

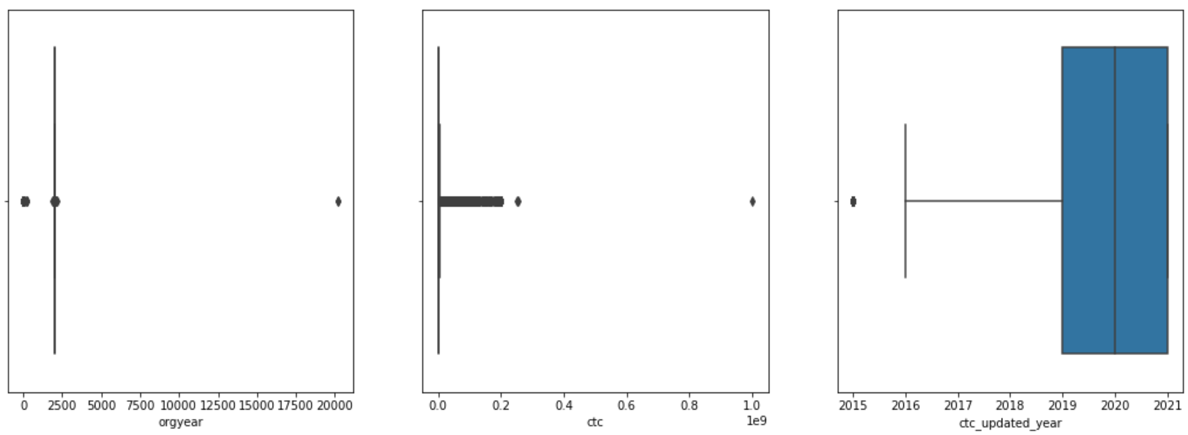
Out[9]:

	company_hash	job_position
count	205799	153281
unique	37299	1017
top	nvnv wgzohrnvzwj otqcxwto	Backend Engineer
freq	8337	43554

- Backend Engineer is the most common job_position

Univariate

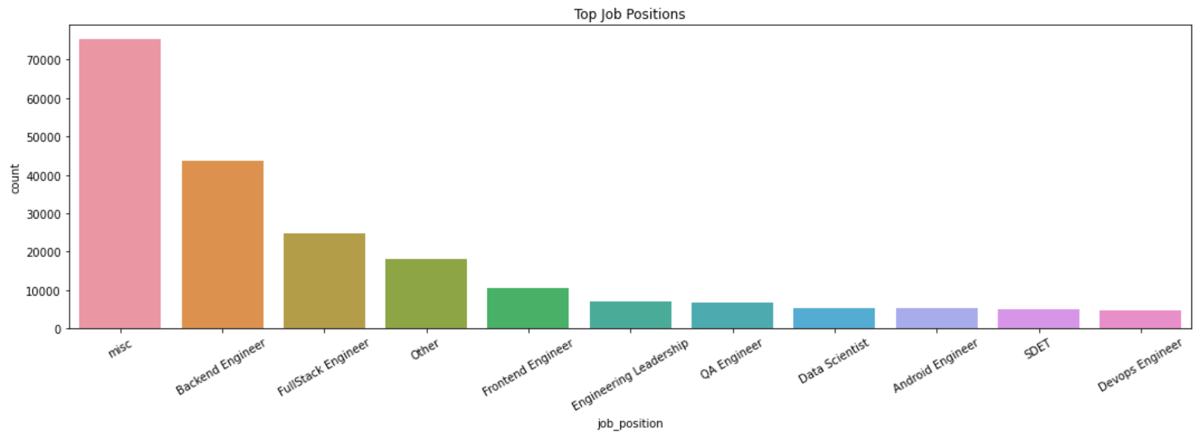
In [10]:
```
fig,ax = plt.subplots(1,3, figsize=(18,6))
for e,c in enumerate(['orgyear','ctc','ctc_updated_year']):
    sns.boxplot(x = df[c], ax=ax[e])
```



- orgyear and ctc has clear outliers
- ctc_updated_year is mostly skewed towards current year

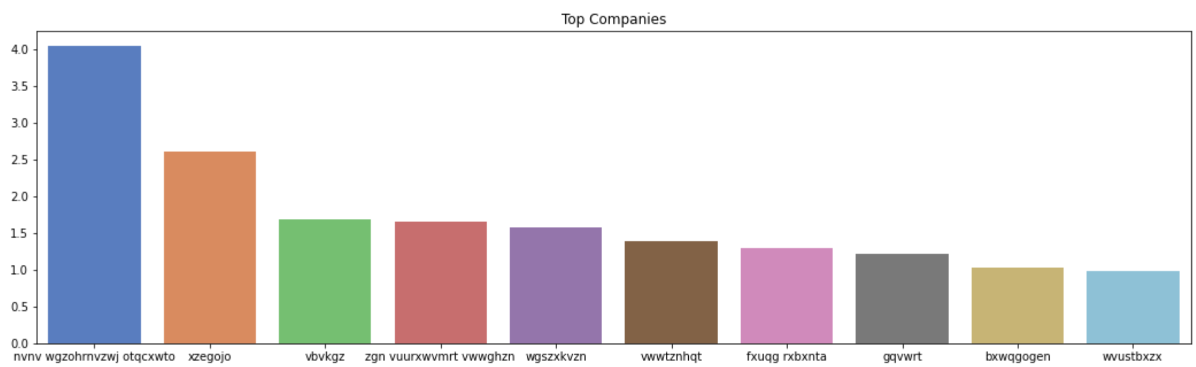
In [11]:
```
tmp = df['job_position'].copy()

top_10_jobs = list(tmp.value_counts().index[:10])

tmp[~tmp.isin(top_10_jobs)] = 'misc'
```

In [12]:
```
plt.rcParams['figure.figsize'] = (18,5)
g = sns.countplot(x=tmp, order=['misc']+top_10_jobs)
g.set_xticklabels(g.get_xticklabels(), rotation=30)
g.set_title('Top Job Positions')
print()
```



- Backend/Fullstack/Frontend Engineers are the most common job positions

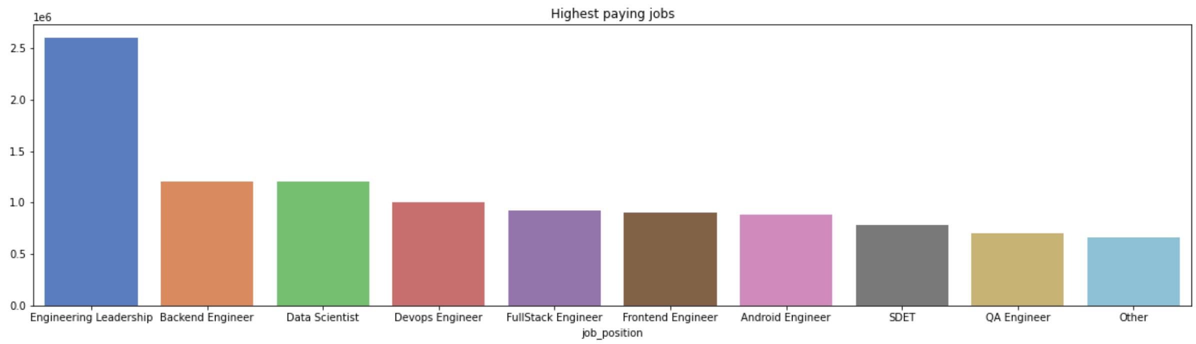
In [13]:
```
tmp = (df['company_hash'].value_counts(1) * 100).head(10)
g = sns.barplot(x=tmp.index, y=tmp.values, palette='muted')
g.set_title('Top Companies')
print()
```



- nvnv wgzohrnvzwj otqcxwto is the most common company ~ 4%

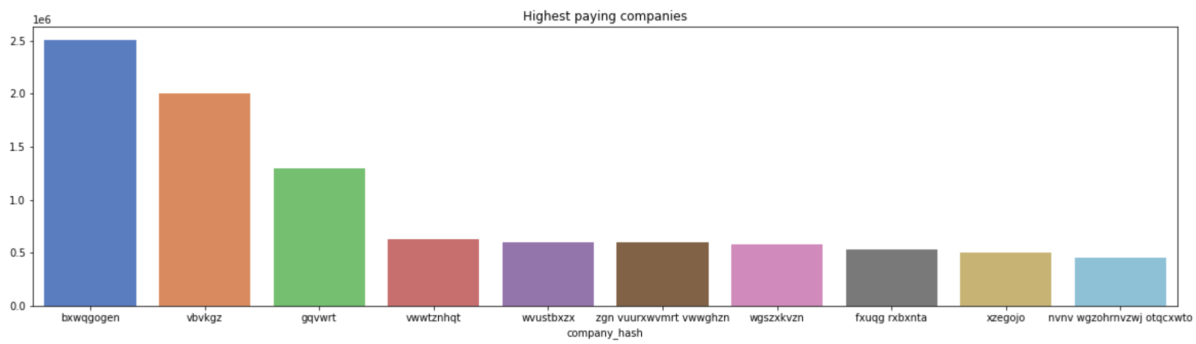
Bivariate

In [14]:
```
plt.rcParams['figure.figsize'] = (20,5)
tmp = df[df['job_position'].isin(top_10_jobs)].groupby('job_position')['ctc'].media
g = sns.barplot(x=tmp.index, y=tmp.values, palette='muted')
g.set_title('Highest paying jobs')
print()
```



- Engineering Leadership is the highest paid job ~ 26Lakh median salary

In [15]:
```
top_10_company = list(df['company_hash'].value_counts().index[:10])
tmp = df[df['company_hash'].isin(top_10_company)].groupby('company_hash')['ctc'].me
g = sns.barplot(x=tmp.index, y=tmp.values, palette='muted')
g.set_title('Highest paying companies')
print()
```

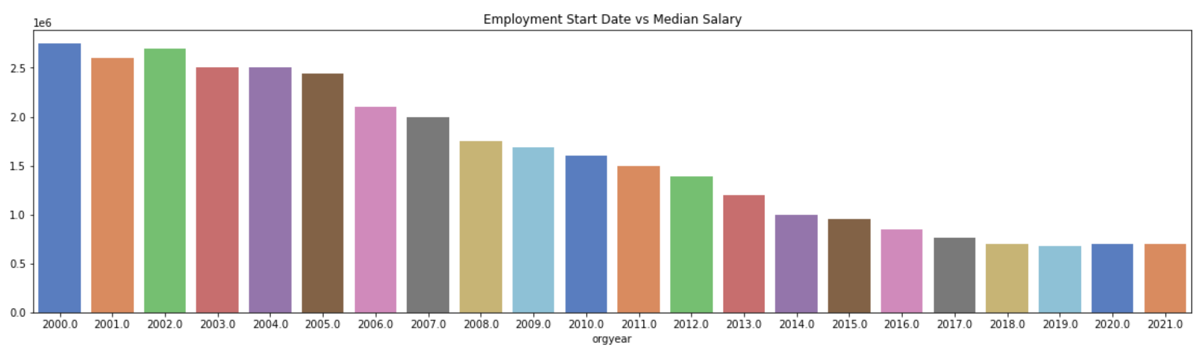


- bxwqgogen is the company which has highest median salary ~ 26Lakh

In [16]:
```
# Subsetting for top 10 jobs in top 10 companies
tmp = df[df['job_position'].isin(top_10_jobs) & df['company_hash'].isin(top_10_comp

tmp = pd.DataFrame(tmp.groupby(['job_position','company_hash'])['ctc'].max())

a = tmp.groupby(level=0)['ctc'].max().values
b = tmp.groupby(level=0)['ctc'].idxmax().values
for i,j in zip(a,b):
    print(f'For {j[0]} job position, \'{j[1]}\' offers the highest package of : Rs
```

```
For Android Engineer job position, 'zgn vuurxwvmrt vwwghzn' offers the highest pac
kage of : Rs 10Cr
For Backend Engineer job position, 'fxuqg rxbxnta' offers the highest package of :
Rs 20Cr
For Data Scientist job position, 'zgn vuurxwvmrt vwwghzn' offers the highest packa
ge of : Rs 9Cr
For Devops Engineer job position, 'vwwtznhqt' offers the highest package of : Rs 9
Cr
For Engineering Leadership job position, 'fxuqg rxbxnta' offers the highest packag
e of : Rs 20Cr
For Frontend Engineer job position, 'bxwqgogen' offers the highest package of : Rs
20Cr
For FullStack Engineer job position, 'fxuqg rxbxnta' offers the highest package of
: Rs 19Cr
For Other job position, 'fxuqg rxbxnta' offers the highest package of : Rs 20Cr
For QA Engineer job position, 'vbvkgz' offers the highest package of : Rs 20Cr
For SDET job position, 'wgszxkvzn' offers the highest package of : Rs 10Cr
```

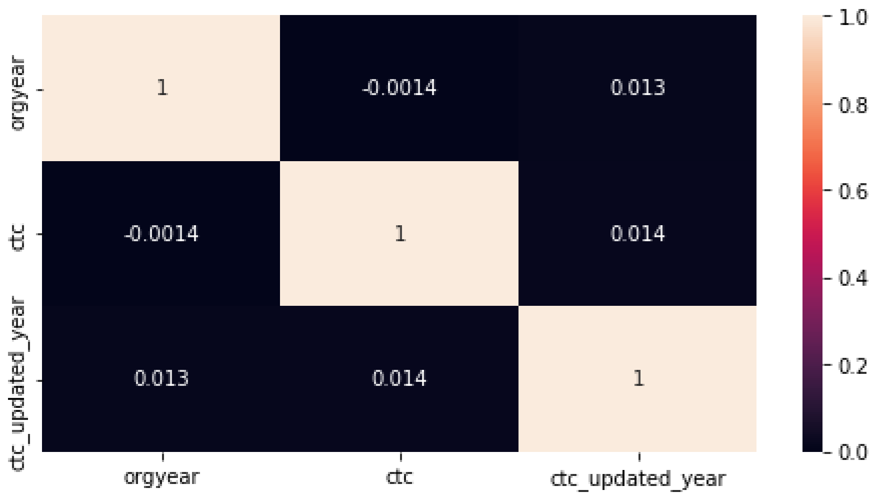
In [17]:
```python
# Subsetting for 2000 - 2022 years
tmp = df[df['orgyear'].isin(np.arange(2000,2022))].copy()
tmp = tmp.groupby('orgyear')['ctc'].median()
g = sns.barplot(x=tmp.index, y=tmp.values, palette='muted')
g.set_title('Employment Start Date vs Median Salary')
print()
```



- There is very linear relationship between number of years spent at an organization vs their salary
- Median salary for employees who joined in
  - 2000 -> 26 Lakh
  - 2021 -> 6 Lakh

In [18]:
```python
plt.rcParams['figure.figsize'] = (8,4)
sns.heatmap(df.corr(), annot=True)
```

Out[18]: <AxesSubplot:>



- No real correlation observed in these fields

Data Pre-processing

Removing outliers

In [19]:
```
def remove_outliers(df, c):
    before = df.shape[0]
    q1 = df[c].quantile(0.25)
    q3 = df[c].quantile(0.75)
    iqr = q3 - q1
    minima = q1 - 1.5*iqr
    maxima = q3 + 1.5*iqr
    df = df[(df[c] >= minima) & (df[c] <= maxima)]
    after = df.shape[0]
    print(f'Before : {before}\tAfter : {after}\tRows Dropped : {before-after}')
    return df.reset_index(drop=True)
```

In [20]:
```
for c in ['orgyear','ctc','ctc_updated_year']:
    print(f'\nColumn : {c}')
    df = remove_outliers(df, c)
```

```
Column : orgyear
Before : 205843 After : 197993  Rows Dropped : 7850

Column : ctc
Before : 197993 After : 185635  Rows Dropped : 12358

Column : ctc_updated_year
Before : 185635 After : 183026  Rows Dropped : 2609
```

Sanitizing columns

In [21]:
```
for c in ['company_hash','job_position']:
    print(f'\nColumn : {c}\nNunique Before: {df[c].nunique(dropna=False)}')
    df[c] = df[c].apply(lambda x : str(x).lower().strip())
    print(f'Nunique After: {df[c].nunique(dropna=False)}')
```

```
Column : company_hash
Nunique Before: 33796
Nunique After: 33796

Column : job_position
Nunique Before: 873
Nunique After: 782
```

Imputation

In [22]:
```
df.isna().sum()
```

Out[22]:
```
company_hash        0
orgyear             0
ctc                 0
job_position        0
ctc_updated_year    0
dtype: int64
```

- Null values got removed from outlier removal

In [23]:
```
df['orgyear'] = df['orgyear'].astype(int)
df['ctc_updated_year'] = df['ctc_updated_year'].astype(int)
```

In [24]:
```
# job position still has 26% nan values
(df['job_position'] == 'nan').mean() * 100
```

Out[24]:
```
26.34762274212407
```

- Here 'nan' acts as a new category in itself. So impuation of categorical feature is not required

In [25]:
```
#Adding new feature
df['experience'] = (2022 - df['orgyear']).astype(int)
```

In [26]:
```
df.head()
```

Out[26]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	experience
0	atrgxnnt xzaxv	2016	1100000	other	2020	6
1	qtrxvzwt xzegwgbb rxbxnta	2018	449999	fullstack engineer	2019	4
2	ojzwnvwnxw vx	2015	2000000	backend engineer	2020	7
3	ngpgutaxv	2017	700000	backend engineer	2019	5
4	qxen sqghu	2017	1400000	fullstack engineer	2019	5

Clubbing low frequency categories into a new one

In [27]:
```
df.shape[0] * 0.005 # Each category should have atleast 0.5% data
```

Out[27]: 915.13

In [28]:
```
tmp = df['job_position'].value_counts(normalize=True)
top_jobs = list(tmp[tmp > 0.005].index)

tmp = df['company_hash'].value_counts(normalize=True)
top_companies = list(tmp[tmp > 0.005].index)
```

In [29]:
```
df.loc[~df['job_position'].isin(top_jobs),'job_position'] = 'misc_job'
df.loc[~df['company_hash'].isin(top_companies),'company_hash'] = 'misc_company'
```

In [30]:
```
df.head()
```

Out[30]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	experience
0	misc_company	2016	1100000	other	2020	6
1	misc_company	2018	449999	fullstack engineer	2019	4
2	misc_company	2015	2000000	backend engineer	2020	7
3	misc_company	2017	700000	backend engineer	2019	5
4	misc_company	2017	1400000	fullstack engineer	2019	5

Removing rows with less than 3 unique cases

- We need to have atleast 3 unique values in each category to create 1,2,3 flags for class, tier and designation

In [31]:
```
df.shape
```

Out[31]: (183026, 6)

In [32]:
```
df['concat'] = df['company_hash'] + '__' + df['job_position'] + '__' + df['experien

tmp = df['concat'].value_counts()

df = df[df['concat'].isin(tmp[tmp > 2].index)]
df = df.drop(columns=['concat'])
```

In [33]:
```
df.shape
```

Out[33]:
```
(181301, 6)
```

In [34]:
```
# adding some noise to ctc so that pd.qcut can find three categories
rng = np.random.RandomState(42)
df['ctc'] = df['ctc'] + rng.normal(size=df.shape[0])
```

- Standardization and Encoding will be done post manual clustering

Manual Clustering

In [35]:
```
df['tier_flag'] = df.groupby('company_hash')['ctc'].transform(lambda x : pd.qcut(x,

df['class_flag'] = df.groupby(['company_hash','job_position'])['ctc'].transform(lam

df['designation_flag'] = df.groupby(['company_hash','job_position','experience'])['
```

In [36]:
```
df.head()
```

Out[36]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	experience	tier_flag	cl
0	misc_company	2016	1.100000e+06	other	2020	6	Middle	
1	misc_company	2018	4.499989e+05	fullstack engineer	2019	4	Bottom	
2	misc_company	2015	2.000001e+06	backend engineer	2020	7	Top	
3	misc_company	2017	7.000015e+05	backend engineer	2019	5	Middle	
4	misc_company	2017	1.400000e+06	fullstack engineer	2019	5	Top	

In [37]:
```
COMPANY_NAME = 'bxwqgogen'
JOB_NAME = 'data scientist'
EXP = [5,6,7]
```

In [38]:
```
print('\n*************TIER LEVEL INSIGHTS**************')
```

```
print(f'\nTop 10 employees at  {COMPANY_NAME}: ')
display(df[df['company_hash'] == COMPANY_NAME].sort_values('ctc',ascending=False).h

print(f'\nBottom 10 employees at {COMPANY_NAME} : ')
display(df[df['company_hash'] == COMPANY_NAME].sort_values('ctc',ascending=False).t
```

```
**************TIER LEVEL INSIGHTS**************

Top 10 employees at  bxwqgogen:
```

	company_hash	orgyear	ctc	job_position	ctc_updated_year	experience	tier_fl
118236	bxwqgogen	2014	3.220001e+06	fullstack engineer	2021	8	T
170196	bxwqgogen	2014	3.220001e+06	backend engineer	2021	8	T
53934	bxwqgogen	2016	3.200002e+06	backend engineer	2020	6	T
105602	bxwqgogen	2018	3.200002e+06	nan	2018	4	T
57679	bxwqgogen	2013	3.200001e+06	backend engineer	2020	9	T
175226	bxwqgogen	2011	3.200001e+06	nan	2019	11	T
76142	bxwqgogen	2011	3.200001e+06	backend engineer	2019	11	T
136091	bxwqgogen	2016	3.200001e+06	backend engineer	2020	6	T
90774	bxwqgogen	2017	3.200001e+06	backend engineer	2019	5	T
165892	bxwqgogen	2015	3.200001e+06	nan	2019	7	T

```
Bottom 10 employees at bxwqgogen :
```

	company_hash	orgyear	ctc	job_position	ctc_updated_year	experience	tier_fla
141217	bxwqgogen	2012	53001.506354	backend engineer	2019	10	Botto
79926	bxwqgogen	2007	35999.425405	backend engineer	2019	15	Botto
90267	bxwqgogen	2015	25000.453366	nan	2019	7	Botto
59573	bxwqgogen	2015	24999.166213	other	2019	7	Botto
13886	bxwqgogen	2011	24001.682888	backend engineer	2019	11	Botto
151890	bxwqgogen	2012	20999.589977	backend engineer	2019	10	Botto
54809	bxwqgogen	2015	17999.485433	nan	2019	7	Botto
117065	bxwqgogen	2015	17999.404821	fullstack engineer	2019	7	Botto
146578	bxwqgogen	2020	9000.594542	support engineer	2020	2	Botto
75771	bxwqgogen	2017	5001.265139	backend engineer	2020	5	Botto

In [39]:
```
print('\n*************CLASS LEVEL INSIGHTS*************')

print(f'\nTop 10 {JOB_NAME} at  {COMPANY_NAME}: ')
display(df[(df['company_hash'] == COMPANY_NAME) & (df['job_position'] == JOB_NAME)]

print(f'\nBottom 10 {JOB_NAME} at {COMPANY_NAME} : ')
display(df[(df['company_hash'] == COMPANY_NAME) & (df['job_position'] == JOB_NAME)]
```

```
*************CLASS LEVEL INSIGHTS*************

Top 10 data scientist at  bxwqgogen:
```

	company_hash	orgyear	ctc	job_position	ctc_updated_year	experience	tier_fl
140312	bxwqgogen	2019	3.099999e+06	data scientist	2021	3	T
146031	bxwqgogen	2018	3.000001e+06	data scientist	2019	4	T
30118	bxwqgogen	2013	3.000000e+06	data scientist	2021	9	T
108184	bxwqgogen	2013	2.999999e+06	data scientist	2021	9	T
27852	bxwqgogen	2016	2.799999e+06	data scientist	2019	6	T
139661	bxwqgogen	2018	2.599999e+06	data scientist	2019	4	T
181502	bxwqgogen	2019	2.500003e+06	data scientist	2020	3	T
110566	bxwqgogen	2017	2.200000e+06	data scientist	2019	5	Midc
169445	bxwqgogen	2018	2.099999e+06	data scientist	2019	4	Midc
140502	bxwqgogen	2017	1.300000e+06	data scientist	2016	5	Botto

```
Bottom 10 data scientist at bxwqgogen :
```

	company_hash	orgyear	ctc	job_position	ctc_updated_year	experience	tier_fl
110566	bxwqgogen	2017	2.200000e+06	data scientist	2019	5	Midc
169445	bxwqgogen	2018	2.099999e+06	data scientist	2019	4	Midc
140502	bxwqgogen	2017	1.300000e+06	data scientist	2016	5	Bottc
43684	bxwqgogen	2017	1.200000e+06	data scientist	2019	5	Bottc
118899	bxwqgogen	2016	7.000010e+05	data scientist	2017	6	Bottc
115704	bxwqgogen	2018	6.999995e+05	data scientist	2016	4	Bottc
130531	bxwqgogen	2018	6.099997e+05	data scientist	2019	4	Bottc
114464	bxwqgogen	2019	4.999993e+05	data scientist	2021	3	Bottc
89949	bxwqgogen	2013	1.999984e+05	data scientist	2020	9	Bottc
167901	bxwqgogen	2016	9.400044e+04	data scientist	2019	6	Bottc

In [40]:

```
print('\n*************DESIGNATION LEVEL INSIGHTS*************')

print(f'\nTop 10 {JOB_NAME} at  {COMPANY_NAME} with {EXP} yrs of experience: ')
display(df[(df['company_hash'] == COMPANY_NAME) & (df['job_position'] == JOB_NAME)

print(f'\nBottom 10 {JOB_NAME} at {COMPANY_NAME} with {EXP} yrs of experience: ')
display(df[(df['company_hash'] == COMPANY_NAME) & (df['job_position'] == JOB_NAME)
```

```
*************DESIGNATION LEVEL INSIGHTS*************

Top 10 data scientist at  bxwqgogen with [5, 6, 7] yrs of experience:
```

	company_hash	orgyear	ctc	job_position	ctc_updated_year	experience	tier_fl
27852	bxwqgogen	2016	2.799999e+06	data scientist	2019	6	T
110566	bxwqgogen	2017	2.200000e+06	data scientist	2019	5	Midc
140502	bxwqgogen	2017	1.300000e+06	data scientist	2016	5	Bottc
43684	bxwqgogen	2017	1.200000e+06	data scientist	2019	5	Bottc
118899	bxwqgogen	2016	7.000010e+05	data scientist	2017	6	Bottc
167901	bxwqgogen	2016	9.400044e+04	data scientist	2019	6	Bottc

```
Bottom 10 data scientist at bxwqgogen with [5, 6, 7] yrs of experience:
```

	company_hash	orgyear	ctc	job_position	ctc_updated_year	experience	tier_fl
27852	bxwqgogen	2016	2.799999e+06	data scientist	2019	6	T
110566	bxwqgogen	2017	2.200000e+06	data scientist	2019	5	Midc
140502	bxwqgogen	2017	1.300000e+06	data scientist	2016	5	Bottc
43684	bxwqgogen	2017	1.200000e+06	data scientist	2019	5	Bottc
118899	bxwqgogen	2016	7.000010e+05	data scientist	2017	6	Bottc
167901	bxwqgogen	2016	9.400044e+04	data scientist	2019	6	Bottc

Unsupervised learning

Encoding and standardization

In [41]:
```
# Label encoding for flags
for c in ['tier_flag', 'class_flag', 'designation_flag']:
    df[c] = df[c].replace({'Top':1, 'Middle':2, 'Bottom':3}).astype(int)
```

In [42]:
```
df = df.reset_index(drop=True)
```

In [43]:
```
df.head()
```

Out[43]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	experience	tier_flag	cl
0	misc_company	2016	1.100000e+06	other	2020	6	2	
1	misc_company	2018	4.499989e+05	fullstack engineer	2019	4	3	
2	misc_company	2015	2.000001e+06	backend engineer	2020	7	1	
3	misc_company	2017	7.000015e+05	backend engineer	2019	5	2	
4	misc_company	2017	1.400000e+06	fullstack engineer	2019	5	1	

One hot encoding of cateogorical columns

In [44]:
```
ohe = OneHotEncoder(sparse=False)
tmp = pd.DataFrame(ohe.fit_transform(df[['company_hash','job_position']]))
df_transformed = pd.concat([df, tmp], axis=1).copy()
df_transformed = df_transformed.drop(columns=['company_hash','job_position'])
```

In [45]:
```
df_transformed.shape
```

Out[45]:
```
(181301, 44)
```

In [46]:
```
df_transformed.head()
```

Out[46]:

	orgyear	ctc	ctc_updated_year	experience	tier_flag	class_flag	designation_flag	0
0	2016	1.100000e+06	2020	6	2	1	1	0.0
1	2018	4.499989e+05	2019	4	3	3	3	0.0
2	2015	2.000001e+06	2020	7	1	1	1	0.0
3	2017	7.000015e+05	2019	5	2	3	3	0.0
4	2017	1.400000e+06	2019	5	1	1	1	0.0

Hopkins Test : Checking clustering tendency

In [47]:
```
hopkins(df_transformed.values, sampling_size=10000)
```

Out[47]:
```
0.006623056396138925
```

- Hopkins score near to 0 means there is strong clustering tendency in the data

Scaling and reducing dimenions for better clustering

In [48]:
```
pipe = make_pipeline(StandardScaler(), PCA(n_components=3, random_state=42))
df_transformed = pd.DataFrame(pipe.fit_transform(df_transformed),columns=['PCA_'+st
```

```
C:\Users\rauna\Anaconda3\envs\scaler\lib\site-packages\sklearn\utils\validation.p
y:1858: FutureWarning: Feature names only support names that are all strings. Got
feature names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
C:\Users\rauna\Anaconda3\envs\scaler\lib\site-packages\sklearn\utils\validation.p
y:1858: FutureWarning: Feature names only support names that are all strings. Got
feature names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

In [49]:
```
df_transformed.head()
```

Out[49]:

	PCA_1	PCA_2	PCA_3
0	0.922862	0.574467	-0.682381
1	-2.253808	-0.925748	-1.005500
2	2.602268	0.403416	-0.737439
3	-1.161687	-1.093157	-0.542895
4	1.745463	0.792766	-1.349309

In [50]:
```
hopkins(df_transformed.values, sampling_size=10000)
```

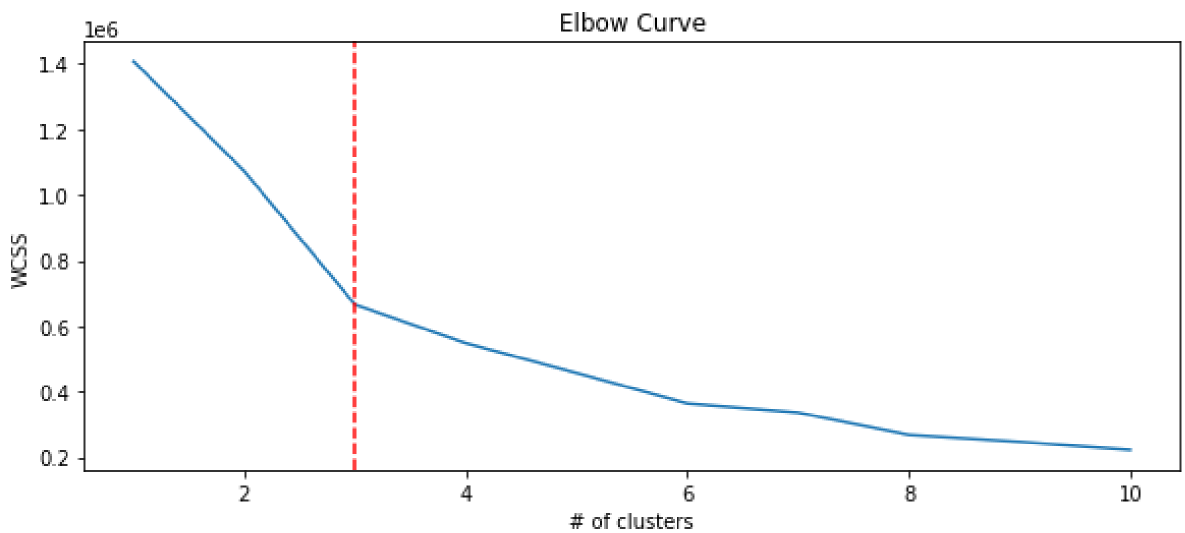
Out[50]:
```
0.010137522966458609
```

Kmeans

Elbow Method

In [51]:
```
plt.rcParams['figure.figsize'] = 10,4
clusters = list(range(1,11))
wcss = []
for k in clusters:
    model = MiniBatchKMeans(n_clusters=k, random_state=42, batch_size=3072)
    model.fit(df_transformed)
    wcss.append(model.inertia_)

g = sns.lineplot(x=clusters, y=wcss)
g.set_title('Elbow Curve')
g.set_xlabel('# of clusters')
g.set_ylabel('WCSS')
g.axvline(3, ls='--', c='r')
```

Out[51]:
```
<matplotlib.lines.Line2D at 0x25f2f837220>
```



- Seems like 3 clusters would make sense

In [52]:
```
model = MiniBatchKMeans(n_clusters=3, random_state=42, batch_size=3072)
model.fit(df_transformed)
```

Out[52]:
```
▾ MiniBatchKMeans
MiniBatchKMeans(batch_size=3072, n_clusters=3, random_state=42)
```

In [53]:
```
df_transformed['clusters'] = model.labels_
```

In [54]:
```
round(df_transformed['clusters'].value_counts(1) * 100)
```

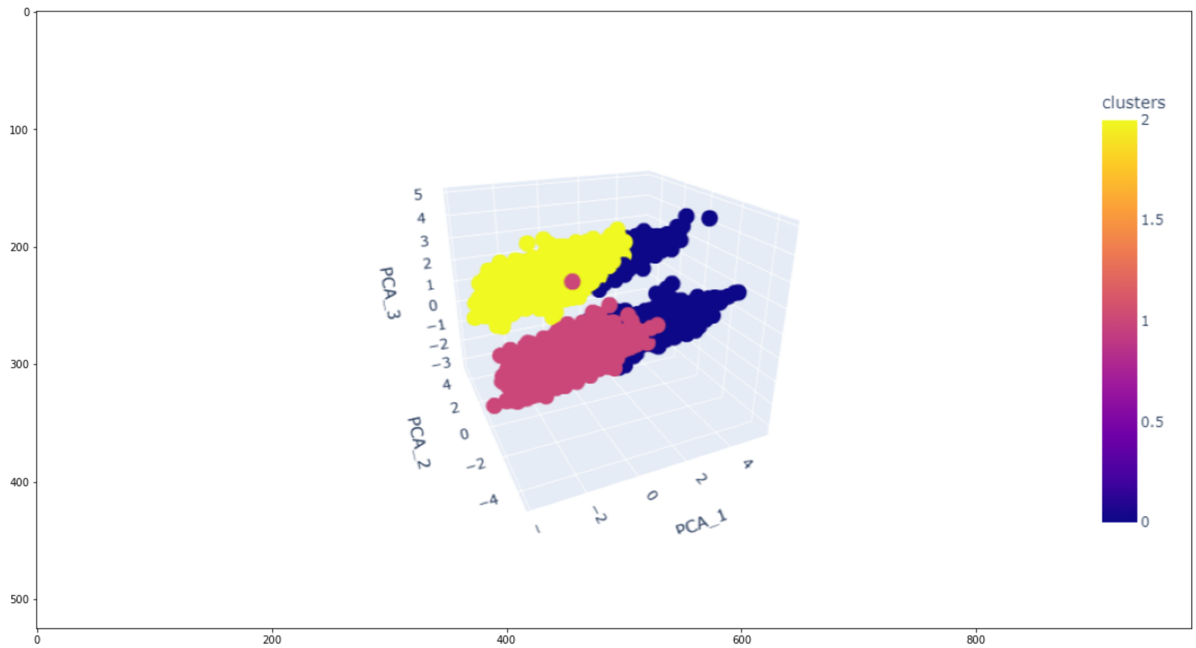
Out[54]:
```
1    48.0
0    32.0
2    20.0
Name: clusters, dtype: float64
```

- 50% points fall in 1 cluster
- 30-20 is the split amongst the other 2 clusters

In [55]:
```
px.scatter_3d(df_transformed.sample(5000), x='PCA_1', y='PCA_2', z='PCA_3', color='
```

In [56]:
```
# Display above image using png
plt.rcParams['figure.figsize'] = 20,20
plt.imshow(mpimg.imread("plot1.png"))
```

Out[56]: <matplotlib.image.AxesImage at 0x25f36b55fd0>



- Looking at the above graph it seems like there should be 2 clusters instead of 3

Using 2 clusters for kmeans

In [57]:
```
model = MiniBatchKMeans(n_clusters=2, random_state=42, batch_size=3072)
model.fit(df_transformed.drop(columns=['clusters']))
```

Out[57]:
```
▾                    MiniBatchKMeans
MiniBatchKMeans(batch_size=3072, n_clusters=2, random_state=42)
```

In [58]:
```
df_transformed['clusters'] = model.labels_
df['clusters'] = model.labels_
```

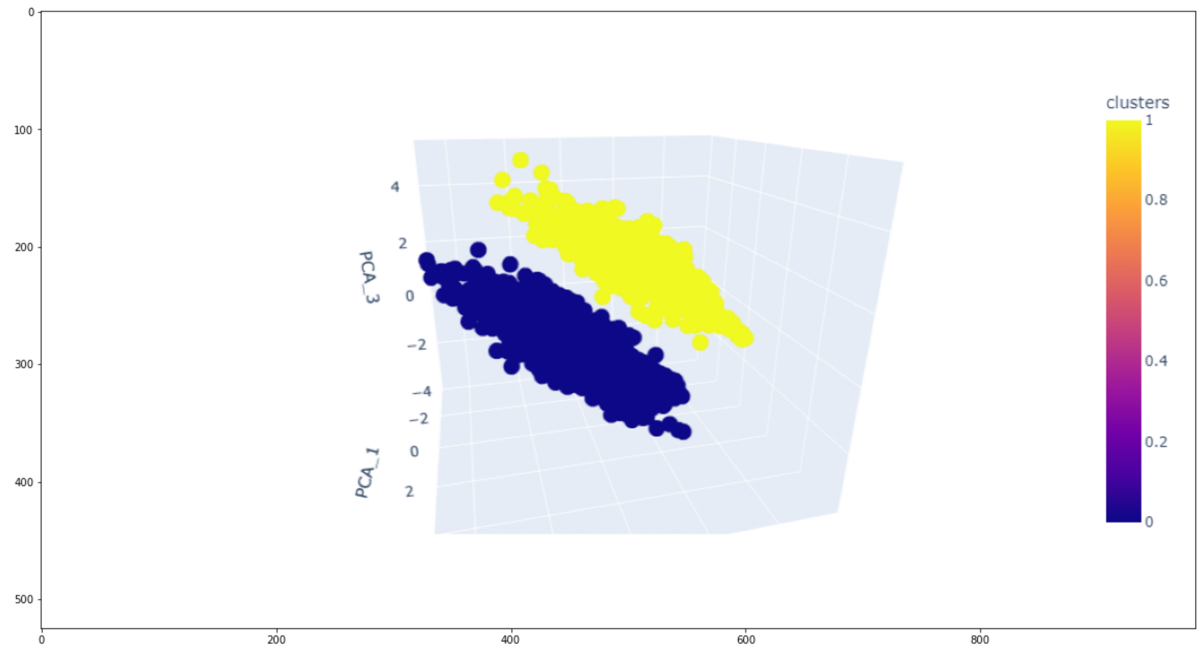
In [59]:
```
round(df_transformed['clusters'].value_counts(1) * 100)
```

Out[59]:
```
0    78.0
1    22.0
Name: clusters, dtype: float64
```

In [60]:
```
px.scatter_3d(df_transformed.sample(5000), x='PCA_1', y='PCA_2', z='PCA_3', color='
```

In [61]:
```python
# Display above image using png
plt.rcParams['figure.figsize'] = 20,20
plt.imshow(mpimg.imread("plot2.png"))
```

Out[61]:
```
<matplotlib.image.AxesImage at 0x25f36a1b7c0>
```



In [62]:
```
model.inertia_
```

Out[62]:
```
1072227.0258263652
```

- These clusters make much more sense now and are clearly seprable

Cluster Visualization

In [63]:
```
df.head()
```

Out[63]:

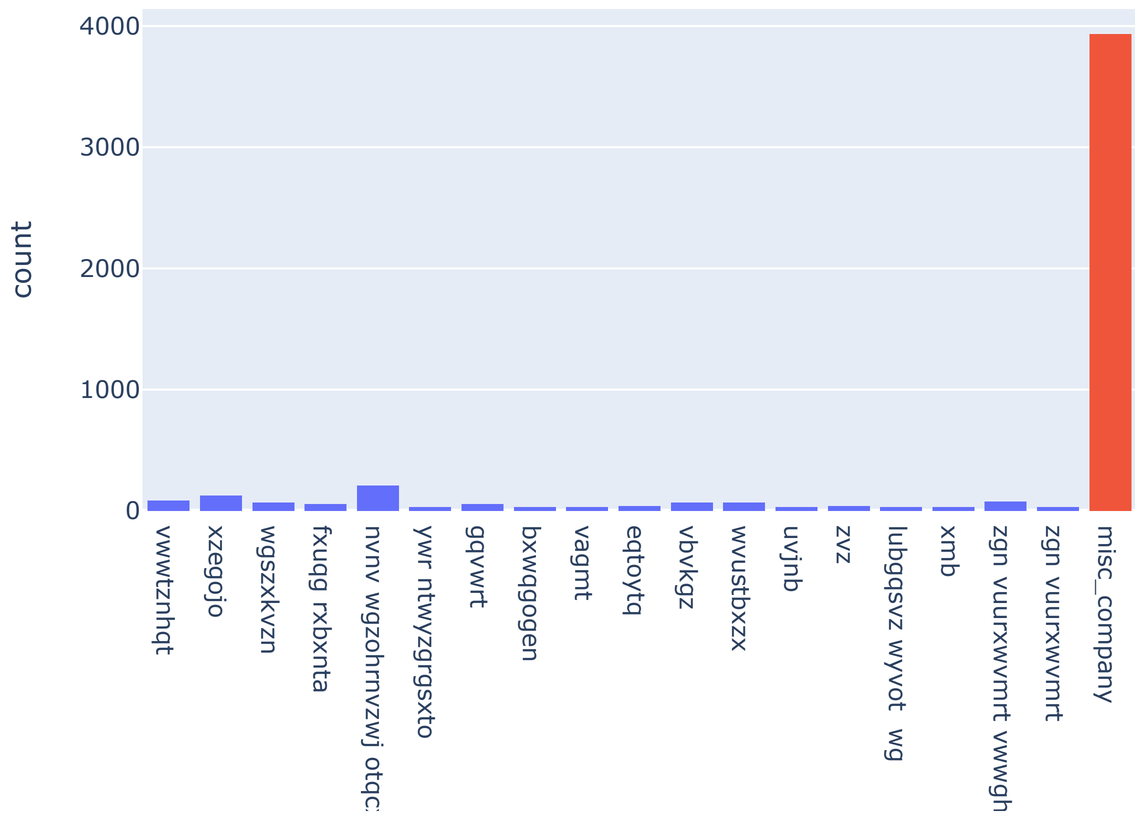
	company_hash	orgyear	ctc	job_position	ctc_updated_year	experience	tier_flag	cl
0	misc_company	2016	1.100000e+06	other	2020	6	2	
1	misc_company	2018	4.499989e+05	fullstack engineer	2019	4	3	
2	misc_company	2015	2.000001e+06	backend engineer	2020	7	1	
3	misc_company	2017	7.000015e+05	backend engineer	2019	5	2	
4	misc_company	2017	1.400000e+06	fullstack engineer	2019	5	1	

In [64]:
```
round(df['clusters'].value_counts(1) * 100)
```

Out[64]:
```
0    78.0
1    22.0
Name: clusters, dtype: float64
```

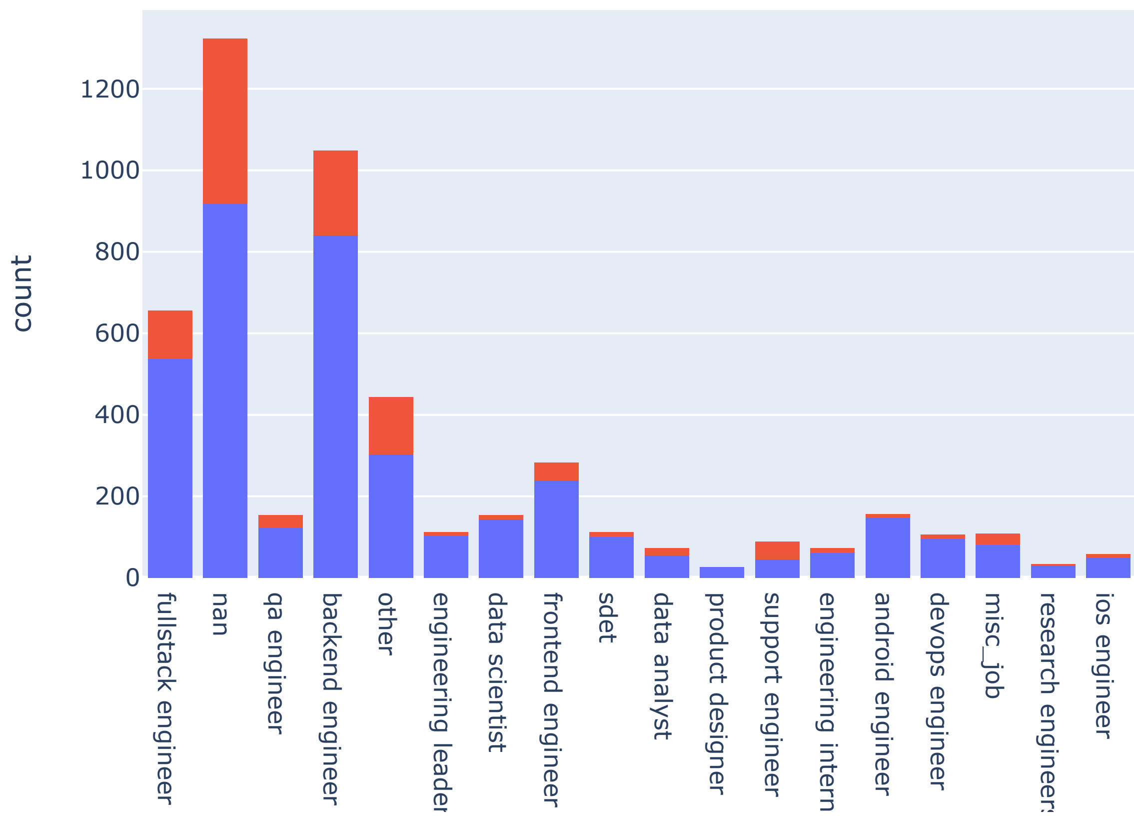
- 80% of data belong to one clusters

```
In [65]: px.histogram(df.sample(5000), x='company_hash', color='clusters')
```



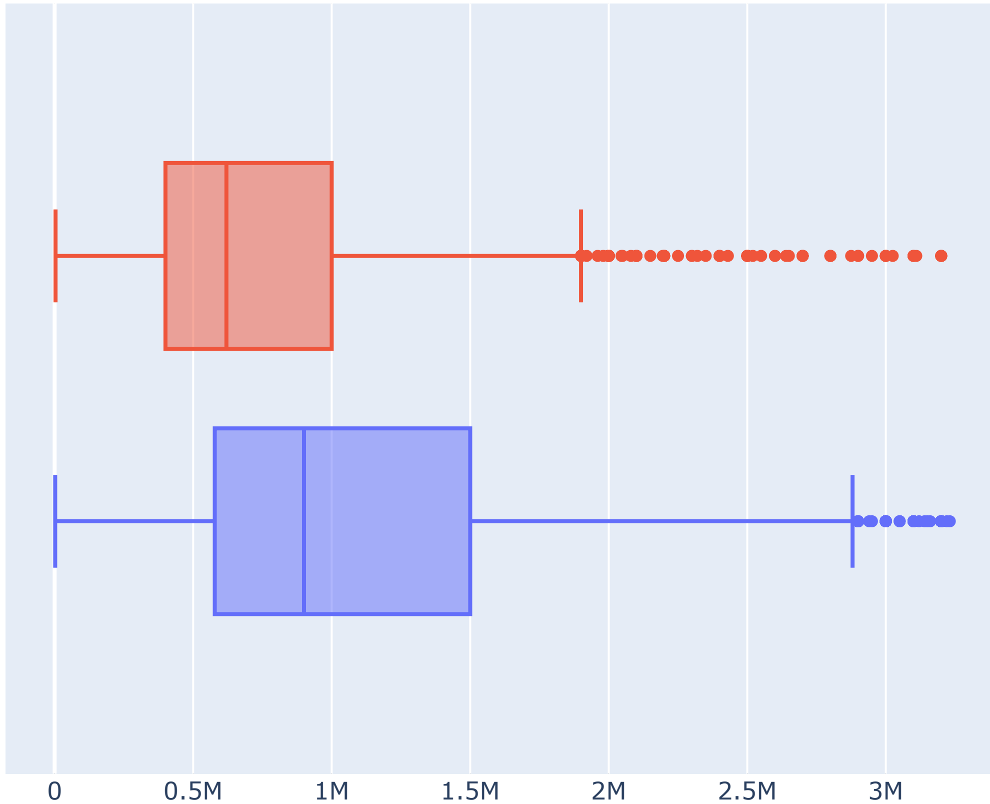
- Company is a clear seperator for these clusters. One cluster of people are from top companies, rest from misc

```
In [66]: px.histogram(df.sample(5000), x='job_position', color='clusters')
```



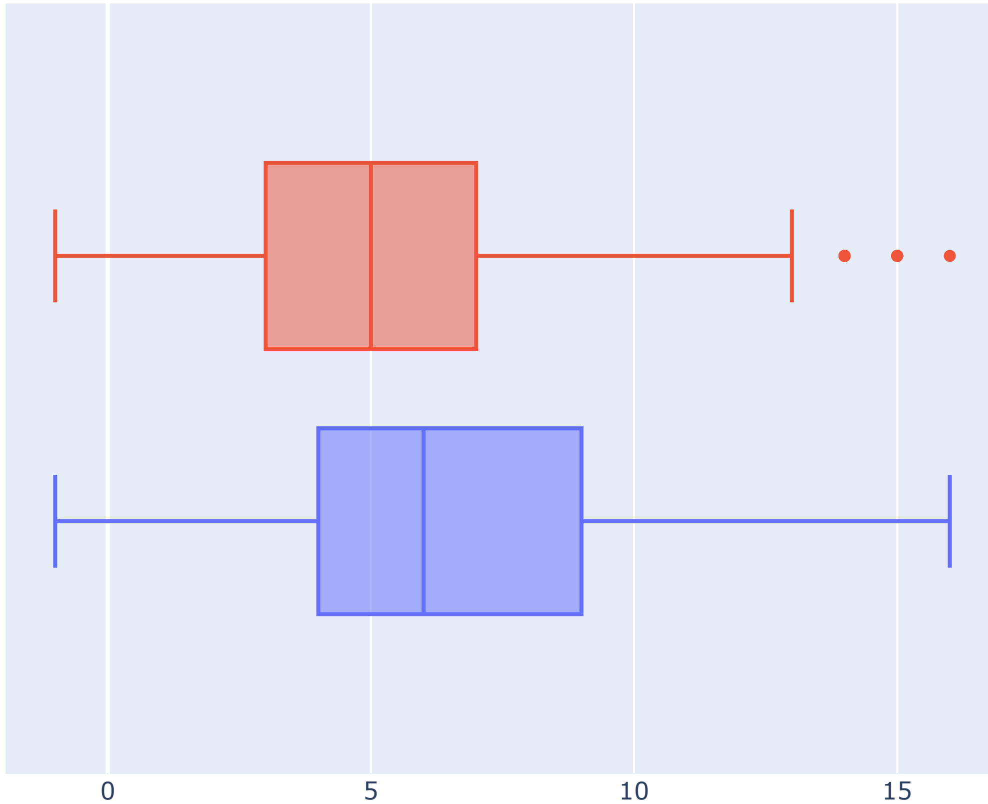
- Minority class are mostly backend/fullstack engineers or have unidentified (other/nan/misc) job positions

In [67]:
```
px.box(df.sample(5000), x='ctc', color='clusters')
```



- The minority cluster is also slightly towards lower end in terms of ctc

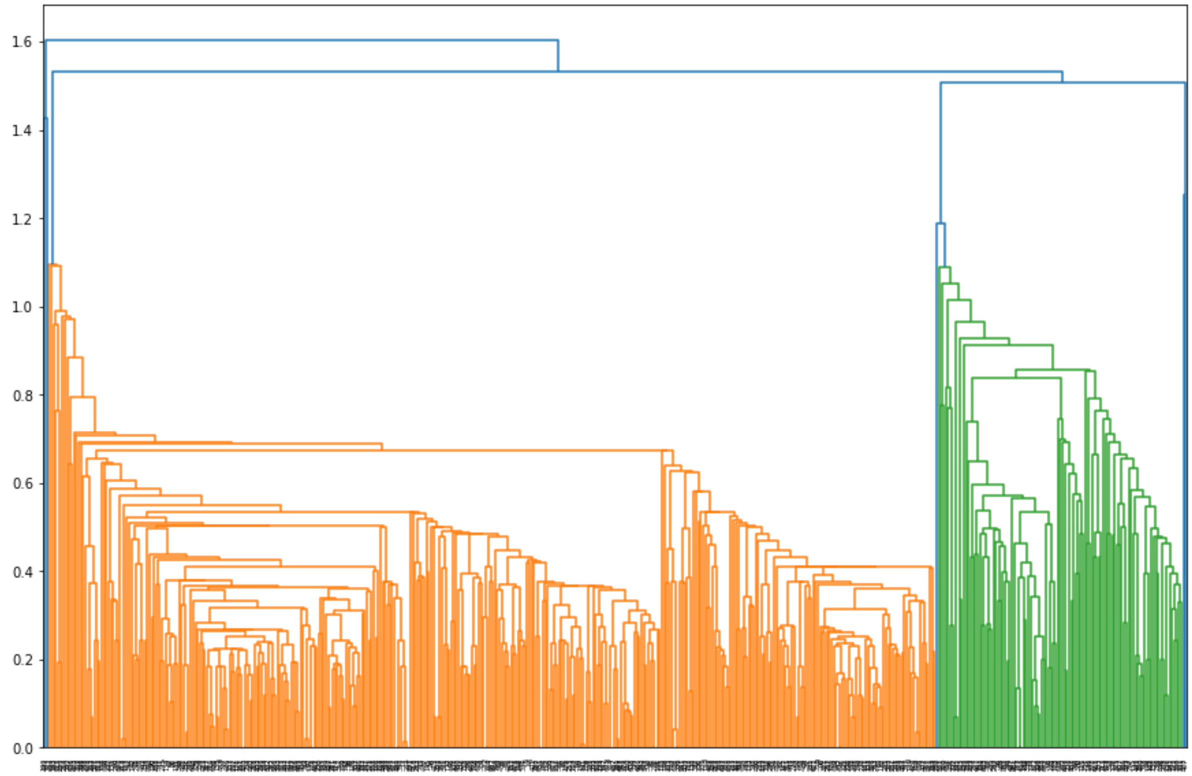
In [68]:
```
px.box(df.sample(5000), x='experience', color='clusters')
```



- The minority cluster is also slightly towards lower end in terms of experience

Hierarichal Clustering

In [69]:
```
# Taking a small sample so that we can visualize
df_transformed_sample = df_transformed.drop(columns='clusters').sample(500, random_

z = linkage(df_transformed_sample)
plt.figure(figsize=(15,10))
results = dendrogram(z)
```



- Dendogram also shows presence of 2 clusters majorly clusters which is inline with our previous clustering techniques
- it also shows somewhat 80-20 split of clusters, also in line with our previous results