Information about the dataset

'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)

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
from collections import Counter
import re
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
#Loading the data.

df = pd.read_csv('https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/002/856/original/scaler_cl
```

In [4]:
```
df.head()
```

Out[4]:

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

In [5]:
```
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      153279 non-null  object 
 6   ctc_updated_year  205843 non-null  float64
dtypes: float64(2), int64(2), object(3)
memory usage: 11.0+ MB
```

In [6]:
```
df.isna().sum()
```

Out[6]:
```
Unnamed: 0             0
company_hash          44
email_hash             0
orgyear               86
ctc                    0
job_position       52564
ctc_updated_year       0
dtype: int64
```

In [7]:
```
def preprocess_string(string):
    """Inputs a string and returns a clean string stripped of any special characters."""
    new_string= re.sub('[^A-Za-z ]+', '', string).lower().strip()
    return new_string

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

Out[7]: 'airtel x labs'

In [8]:
```
#Using the function created above to clean the string in 'job position' feature.

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

Out[8]: 856

In [9]:
```
#Cleaning the strings in 'company_hash' column.

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

Out[9]: 37208

In [10]:
```
df.shape
```

Out[10]: (205843, 7)

In [11]:
```
#Dropping unnecessary "Unnamed: 0" column.

df.drop("Unnamed: 0", inplace=True, axis=1)
```

In [12]:
```
df.head()
```

Out[12]:

	company_hash	email_hash	orgyear	ctc	job_position	ctc_updated_ye
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	1100000	other	2020
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018.0	449999	fullstack engineer	2019
2	ojzwnvwnxw vx	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2015.0	2000000	backend engineer	2020
3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	2017.0	700000	backend engineer	2019
4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	2017.0	1400000	fullstack engineer	2019

In [13]:
```
#Counting duplicates

df.duplicated().sum()
```

Out[13]: 146

In [14]:
```
#Dropping duplicates

df = df.drop_duplicates()
```

In [15]:
```
df.shape
```

Out[15]: (205697, 6)

In [16]:
```
df.isna().sum()
```

Out[16]:
```
company_hash         0
email_hash           0
orgyear             86
ctc                  0
job_position         0
ctc_updated_year     0
dtype: int64
```

In [17]:
```
#Converting string 'nan' to np.nan.

df['job_position'] = df['job_position'].replace('nan', np.nan)
```

In [18]:
```
#Converting string 'nan' to np.nan.

df['company_hash'] = df['company_hash'].replace('nan', np.nan)
```

In [19]:
```
df
```

Out[19]:

	company_hash	email_hash	orgyear	ctc	job_position	ctc_upda
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	1100000	other	
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018.0	449999	fullstack engineer	
2	ojzwnvwnxw vx	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2015.0	2000000	backend engineer	
3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	2017.0	700000	backend engineer	
4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	2017.0	1400000	fullstack engineer	
...	...	...	...	...	...	
205838	vuurt xzw	70027b728c8ee901fe979533ed94ffda97be08fc23f33b...	2008.0	220000	NaN	
205839	husqvawgb	7f7292ffad724ebbe9ca860f515245368d714c84705b42...	2017.0	500000	NaN	
205840	vwwgrxnt	cb25cc7304e9a24facda7f5567c7922ffc48e3d5d6018c...	2021.0	700000	NaN	
205841	zgn vuurxwvmrt	fb46a1a2752f5f652ce634f6178d0578ef6995ee59f6c8...	2019.0	5100000	NaN	
205842	bgqsvz onvzrtj	0bcfc1d05f2e8dc4147743a1313aa70a119b41b30d4a1f...	2014.0	1240000	NaN	

205697 rows × 6 columns

We notice that there are 52564 empty job positions in the data. Here we adopt the following strategy to impute these fields,

1. We first impute the job positions with the most frequently occurring job positions within a company.

2. Then we impute the 'NaN' job positions with those positions whose corresponding 'ctc' is closest to NaN position's ctc.

3. Finally we impute the left over job positions with 'other'

In [21]:
```
def temp(lst):
    """Returns the key of the maximum frequency value."""
    d = Counter(lst)
    return max(d, key=d.get)

#Finding the most frequent job position within the same company and CTC, then imputing 'NaN' job position w

df['job_position'] = df['job_position'].fillna(df.groupby(['company_hash', 'ctc'])['job_position'].transfor
```

In [22]:
```
#We see a reduction in the NaN values in job position.

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

Out[22]:
```
company_hash          44
email_hash             0
orgyear               86
ctc                    0
job_position       37126
ctc_updated_year       0
dtype: int64
```

In [23]:
```
#Filling the companies that have no name with 'other'.

df['company_hash'] = df['company_hash'].replace('', 'other')
```

In [24]:
```
#Creating a temporary dataframe which has columns 'company_hash', 'job_position' and mean ctc for each job

df_temp = df.groupby(['company_hash', 'job_position']).agg('mean', 'ctc').reset_index()
```

In [25]:
```
def impute_job_position(row, avg_ctc_by_position):
    """
    Inputs a row from the original dataframe and 'df_temp'.

    If the job position is 'NaN', it filters 'avg_ctc_by_position' for that company,
    and calculates the closest ctc to the ctc of the 'NaN' job_position

    Returns the corresponding job_postion closest to the 'NaN' ctc.
    Returns the orginal row, if the job_position is not 'NaN'

    """
    if pd.isna(row["job_position"]):  # If job_position is NaN
        company_jobs = avg_ctc_by_position[avg_ctc_by_position["company_hash"] == row["company_hash"]]
        if not company_jobs.empty:
            # Find the job with the closest CTC
            closest_job = company_jobs.iloc[(company_jobs["ctc"] - row["ctc"]).abs().argsort()[:1]]["job_po
            return closest_job
    return row["job_position"]

#Filling the 'NaN' job position with the closest job_position basis the ctc.
df["job_position"] = df.apply(lambda row: impute_job_position(row, df_temp), axis=1)
```

In [26]:
```
#We see a significant decrease in the NaN values.

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

Out[26]:
```
company_hash          44
email_hash             0
orgyear               86
ctc                    0
job_position        4045
ctc_updated_year       0
dtype: int64
```

In [27]:
```
df
```

Out[27]:

	company_hash	email_hash	orgyear	ctc	job_position	ctc_upda
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	1100000	other	
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018.0	449999	fullstack engineer	
2	ojzwnvwnxw vx	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2015.0	2000000	backend engineer	
3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	2017.0	700000	backend engineer	
4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	2017.0	1400000	fullstack engineer	
...	...	...	...	...	...	
205838	vuurt xzw	70027b728c8ee901fe979533ed94ffda97be08fc23f33b...	2008.0	220000	NaN	
205839	husqvawgb	7f7292ffad724ebbe9ca860f515245368d714c84705b42...	2017.0	500000	fullstack engineer	
205840	vwwgrxnt	cb25cc7304e9a24facda7f5567c7922ffc48e3d5d6018c...	2021.0	700000	qa engineer	
205841	zgn vuurxwvmrt	fb46a1a2752f5f652ce634f6178d0578ef6995ee59f6c8...	2019.0	5100000	frontend engineer	
205842	bgqsvz onvzrtj	0bcfc1d05f2e8dc4147743a1313aa70a119b41b30d4a1f...	2014.0	1240000	associate software developer	

205697 rows × 6 columns

In [28]:
```
#Filling the rest of the 'NaN' job_positions with 'other'.

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

In [29]:
```
df.isna().sum()
```

Out[29]:
```
company_hash         0
email_hash           0
orgyear             86
ctc                  0
job_position         0
ctc_updated_year     0
dtype: int64
```

In [30]:
```
#Clubbing the companies that are very less in number(<5) under the name 'Others'.

df.loc[df.groupby('company_hash')['ctc'].transform('count') < 5, 'company_hash'] = 'Others'
```

In [31]:
```
df['company_hash'].nunique()
```

Out[31]:
```
3780
```

In [32]:
```
#Filling the 'NaN' values in the 'orgyear' column with median orgyear of that company.

df['orgyear'].fillna(df.groupby('company_hash')['orgyear'].transform('median'), inplace=True)
```

In [33]:
```
#We finally get rid of all NaN values.

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

Out[33]:
```
company_hash        0
email_hash          0
orgyear             0
ctc                 0
job_position        0
ctc_updated_year    0
dtype: int64
```

In [34]:
```
# Updating 'ctc_updated_year' with the correct year.

df['ctc_updated_year'] = df[['ctc_updated_year','orgyear']].max(axis=1)
```

In [35]:
```
# Dropping rows with irregular year value.

df = df[~(df['ctc_updated_year'] > 2025)]
```

Feature Engineering

We create the following new fields that will help us with clustering.

'years_of_experience' -> current year(2025) - 'orgyear'

Designation -> Learners with CTC greater than the Average of their Company's department having same Years of Experience - Values [1,2,3]

Class -> Above analysis at Company & Job Position level - Values [1,2,3]

Tier -> Repeating the same analysis at the Company level [1,2,3]

In [38]:
```
#Creating a new column 'years_of_experience'.

df['years_of_experience'] = 2025 - df['orgyear']
df
```

Out[38]:

	company_hash	email_hash	orgyear	ctc	job_position	ctc_upda
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	1100000	other	
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018.0	449999	fullstack engineer	
2	Others	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2015.0	2000000	backend engineer	
3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	2017.0	700000	backend engineer	
4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	2017.0	1400000	fullstack engineer	
...	...	...	...	...	...	
205838	vuurt xzw	70027b728c8ee901fe979533ed94ffda97be08fc23f33b...	2008.0	220000	other	
205839	husqvawgb	7f7292ffad724ebbe9ca860f515245368d714c84705b42...	2017.0	500000	fullstack engineer	
205840	vwwgrxnt	cb25cc7304e9a24facda7f5567c7922ffc48e3d5d6018c...	2021.0	700000	qa engineer	
205841	zgn vuurxwvmrt	fb46a1a2752f5f652ce634f6178d0578ef6995ee59f6c8...	2019.0	5100000	frontend engineer	
205842	bgqsvz onvzrtj	0bcfc1d05f2e8dc4147743a1313aa70a119b41b30d4a1f...	2014.0	1240000	associate software developer	

205667 rows × 7 columns

In [39]:
```
#Calculating stats.

df3 = df.groupby(['company_hash', 'job_position', 'years_of_experience']).agg(mean_ctc = ('ctc', 'mean'),
                                                                               median_ctc = ('ctc', 'median'),
                                                                               max_ctc = ('ctc', 'max'),
                                                                               min_ctc = ('ctc', 'min'),
                                                                               count_ctc = ('ctc', 'count'))

df3
```

Out[39]:

company_hash	job_position	years_of_experience	mean_ctc	median_ctc	max_ctc	min_ctc	count_ctc
Others		4.0	4.500000e+05	450000.0	450000	450000	1
		6.0	4.200000e+05	420000.0	420000	420000	1
		7.0	3.500000e+05	350000.0	350000	350000	1
		14.0	7.000000e+05	700000.0	700000	700000	1
	a group chat application	11.0	5.000000e+05	500000.0	500000	500000	1
...	...	...	...	...	...	...	...
zxztrtvuo	member of technical staff at nineleaps	9.0	1.200000e+06	1200000.0	1200000	1200000	1
	other	5.0	4.416667e+05	450000.0	450000	400000	6
		6.0	4.250000e+05	425000.0	450000	400000	4
	software developer intern	9.0	1.200000e+06	1200000.0	1200000	1200000	1
		10.0	1.200000e+06	1200000.0	1200000	1200000	1

66343 rows × 5 columns

In [40]:
```
#Merging the two dataframes for comparison.
df_merged = df.merge(df3, on=['company_hash', 'job_position', 'years_of_experience'], how='left')
df_merged
```

Out[40]:

	company_hash	email_hash	orgyear	ctc	job_position	ctc_upda
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	1100000	other	
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018.0	449999	fullstack engineer	
2	Others	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2015.0	2000000	backend engineer	
3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	2017.0	700000	backend engineer	
4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	2017.0	1400000	fullstack engineer	
...	...	...	...	...	...	
205662	vuurt xzw	70027b728c8ee901fe979533ed94ffda97be08fc23f33b...	2008.0	220000	other	
205663	husqvawgb	7f7292ffad724ebbe9ca860f515245368d714c84705b42...	2017.0	500000	fullstack engineer	
205664	vwwgrxnt	cb25cc7304e9a24facda7f5567c7922ffc48e3d5d6018c...	2021.0	700000	qa engineer	
205665	zgn vuurxwvmrt	fb46a1a2752f5f652ce634f6178d0578ef6995ee59f6c8...	2019.0	5100000	frontend engineer	
205666	bgqsvz onvzrtj	0bcfc1d05f2e8dc4147743a1313aa70a119b41b30d4a1f...	2014.0	1240000	associate software developer	

205667 rows × 12 columns

In [41]:
```
def return_flag(ctc, mean_ctc):
    """Inputs ctc and average ctc, and returns 1, 2 or 3 on the basis of given conditions."""
    if ctc > mean_ctc:
        return 1
    elif ctc == mean_ctc:
        return 2
    else:
        return 3

#Creating 'designation' column.
df_merged['flag_position_level'] = df_merged[['ctc', 'mean_ctc']].apply(lambda x: return_flag(x[0], x[1]),
df_merged
```

Out[41]:

	company_hash	email_hash	orgyear	ctc	job_position	ctc_upda
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	1100000	other	
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018.0	449999	fullstack engineer	
2	Others	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2015.0	2000000	backend engineer	
3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	2017.0	700000	backend engineer	
4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	2017.0	1400000	fullstack engineer	
...	...	...	...	...	...	
205662	vuurt xzw	70027b728c8ee901fe979533ed94ffda97be08fc23f33b...	2008.0	220000	other	
205663	husqvawgb	7f7292ffad724ebbe9ca860f515245368d714c84705b42...	2017.0	500000	fullstack engineer	
205664	vwwgrxnt	cb25cc7304e9a24facda7f5567c7922ffc48e3d5d6018c...	2021.0	700000	qa engineer	
205665	zgn vuurxwvmrt	fb46a1a2752f5f652ce634f6178d0578ef6995ee59f6c8...	2019.0	5100000	frontend engineer	
205666	bgqsvz onvzrtj	0bcfc1d05f2e8dc4147743a1313aa70a119b41b30d4a1f...	2014.0	1240000	associate software developer	

205667 rows × 13 columns

In [42]:
```
#Dropping unnecessary stats.

df_merged.drop(['mean_ctc', 'median_ctc', 'max_ctc', 'min_ctc', 'count_ctc'], inplace=True, axis=1)
```

In [43]:
```
df_merged.rename({'flag_position_level': 'designation'}, inplace=True, axis=1)
```

In [44]:
```
df_merged
```

Out[44]:

	company_hash	email_hash	orgyear	ctc	job_position	ctc_upda
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	1100000	other	
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018.0	449999	fullstack engineer	
2	Others	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2015.0	2000000	backend engineer	
3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	2017.0	700000	backend engineer	
4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	2017.0	1400000	fullstack engineer	
...	...	...	...	...	...	
205662	vuurt xzw	70027b728c8ee901fe979533ed94ffda97be08fc23f33b...	2008.0	220000	other	
205663	husqvawgb	7f7292ffad724ebbe9ca860f515245368d714c84705b42...	2017.0	500000	fullstack engineer	
205664	vwwgrxnt	cb25cc7304e9a24facda7f5567c7922ffc48e3d5d6018c...	2021.0	700000	qa engineer	
205665	zgn vuurxwvmrt	fb46a1a2752f5f652ce634f6178d0578ef6995ee59f6c8...	2019.0	5100000	frontend engineer	
205666	bgqsvz onvzrtj	0bcfc1d05f2e8dc4147743a1313aa70a119b41b30d4a1f...	2014.0	1240000	associate software developer	

205667 rows × 8 columns

In [45]:
```
#Finding mean at company and position level.

df4 = df.groupby(['company_hash', 'job_position']).agg(mean_ctc = ('ctc', 'mean'))
df4
```

Out[45]:

company_hash	job_position	mean_ctc
Others		4.800000e+05
	a group chat application	5.000000e+05
	abap developer	5.000000e+05
	administrative clerk	5.000000e+05
	administrator	1.940000e+06
...	...	...
zxztrtvuo	fullstack engineer	8.507273e+05
	ios engineer	1.237500e+06
	member of technical staff at nineleaps	1.200000e+06
	other	4.350000e+05
	software developer intern	1.200000e+06

22732 rows × 1 columns

In [46]:
```
#Merging for comparison.

df_merged2 = df_merged.merge(df4, on=['company_hash', 'job_position'], how='left')
df_merged2
```

Out[46]:

	company_hash	email_hash	orgyear	ctc	job_position	ctc_upda
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	1100000	other	
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018.0	449999	fullstack engineer	
2	Others	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2015.0	2000000	backend engineer	
3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	2017.0	700000	backend engineer	
4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	2017.0	1400000	fullstack engineer	
...	...	...	...	...	...	
205662	vuurt xzw	70027b728c8ee901fe979533ed94ffda97be08fc23f33b...	2008.0	220000	other	
205663	husqvawgb	7f7292ffad724ebbe9ca860f515245368d714c84705b42...	2017.0	500000	fullstack engineer	
205664	vwwgrxnt	cb25cc7304e9a24facda7f5567c7922ffc48e3d5d6018c...	2021.0	700000	qa engineer	
205665	zgn vuurxwvmrt	fb46a1a2752f5f652ce634f6178d0578ef6995ee59f6c8...	2019.0	5100000	frontend engineer	
205666	bgqsvz onvzrtj	0bcfc1d05f2e8dc4147743a1313aa70a119b41b30d4a1f...	2014.0	1240000	associate software developer	

205667 rows × 9 columns

In [47]:
```
#Creating 'Class' column.

df_merged2['Class'] = df_merged2[['ctc', 'mean_ctc']].apply(lambda x: return_flag(x[0], x[1]), axis=1)
df_merged2
```

Out[47]:

	company_hash	email_hash	orgyear	ctc	job_position	ctc_upda
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	1100000	other	
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018.0	449999	fullstack engineer	
2	Others	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2015.0	2000000	backend engineer	
3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	2017.0	700000	backend engineer	
4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	2017.0	1400000	fullstack engineer	
...	...	...	...	...	...	
205662	vuurt xzw	70027b728c8ee901fe979533ed94ffda97be08fc23f33b...	2008.0	220000	other	
205663	husqvawgb	7f7292ffad724ebbe9ca860f515245368d714c84705b42...	2017.0	500000	fullstack engineer	
205664	vwwgrxnt	cb25cc7304e9a24facda7f5567c7922ffc48e3d5d6018c...	2021.0	700000	qa engineer	
205665	zgn vuurxwvmrt	fb46a1a2752f5f652ce634f6178d0578ef6995ee59f6c8...	2019.0	5100000	frontend engineer	
205666	bgqsvz onvzrtj	0bcfc1d05f2e8dc4147743a1313aa70a119b41b30d4a1f...	2014.0	1240000	associate software developer	

205667 rows × 10 columns

In [48]:
```
df_merged2.drop('mean_ctc', axis=1, inplace=True)
df_merged2
```

Out[48]:

	company_hash	email_hash	orgyear	ctc	job_position	ctc_upda
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	1100000	other	
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018.0	449999	fullstack engineer	
2	Others	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2015.0	2000000	backend engineer	
3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	2017.0	700000	backend engineer	
4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	2017.0	1400000	fullstack engineer	
...	...	...	...	...	...	
205662	vuurt xzw	70027b728c8ee901fe979533ed94ffda97be08fc23f33b...	2008.0	220000	other	
205663	husqvawgb	7f7292ffad724ebbe9ca860f515245368d714c84705b42...	2017.0	500000	fullstack engineer	
205664	vwwgrxnt	cb25cc7304e9a24facda7f5567c7922ffc48e3d5d6018c...	2021.0	700000	qa engineer	
205665	zgn vuurxwvmrt	fb46a1a2752f5f652ce634f6178d0578ef6995ee59f6c8...	2019.0	5100000	frontend engineer	
205666	bgqsvz onvzrtj	0bcfc1d05f2e8dc4147743a1313aa70a119b41b30d4a1f...	2014.0	1240000	associate software developer	

205667 rows × 9 columns

In [49]:
```
#Finding mean on company level.

df5 = df.groupby('company_hash').agg(mean_ctc = ('ctc', 'mean'))
df5
```

Out[49]:

company_hash	mean_ctc
Others	2.592725e+06
a ntwyzgrgsxto	1.234688e+06
aaqxctz avnv owxtzwto vzvrjnxwo ucn rna	9.850000e+05
abwavnv ojontb	7.320000e+05
adw ntwyzgrgsj	1.234200e+06
...	...
zxxn rna	9.014286e+05
zxyxrtzn	6.887500e+05
zxyxrtzn ntwyzgrgsxto	8.170000e+05
zxzlvwvqn	1.739048e+06
zxztrtvuo	1.172701e+06

3780 rows × 1 columns

In [50]:
```
#Merging for comparison.

df_merged3 = df_merged2.merge(df5, on=['company_hash'], how='left')
df_merged3
```

Out[50]:

	company_hash	email_hash	orgyear	ctc	job_position	ctc_upda
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	1100000	other	
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018.0	449999	fullstack engineer	
2	Others	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2015.0	2000000	backend engineer	
3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	2017.0	700000	backend engineer	
4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	2017.0	1400000	fullstack engineer	
...	...	...	...	...	...	
205662	vuurt xzw	70027b728c8ee901fe979533ed94ffda97be08fc23f33b...	2008.0	220000	other	
205663	husqvawgb	7f7292ffad724ebbe9ca860f515245368d714c84705b42...	2017.0	500000	fullstack engineer	
205664	vwwgrxnt	cb25cc7304e9a24facda7f5567c7922ffc48e3d5d6018c...	2021.0	700000	qa engineer	
205665	zgn vuurxwvmrt	fb46a1a2752f5f652ce634f6178d0578ef6995ee59f6c8...	2019.0	5100000	frontend engineer	
205666	bgqsvz onvzrtj	0bcfc1d05f2e8dc4147743a1313aa70a119b41b30d4a1f...	2014.0	1240000	associate software developer	

205667 rows × 10 columns

In [51]:
```
#Creating 'Tier' column.

df_merged3['Tier'] = df_merged3[['ctc', 'mean_ctc']].apply(lambda x: return_flag(x[0], x[1]), axis=1)
df_merged3
```

Out[51]:

	company_hash	email_hash	orgyear	ctc	job_position	ctc_upda
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	1100000	other	
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018.0	449999	fullstack engineer	
2	Others	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2015.0	2000000	backend engineer	
3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	2017.0	700000	backend engineer	
4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	2017.0	1400000	fullstack engineer	
...	...	...	...	...	...	
205662	vuurt xzw	70027b728c8ee901fe979533ed94ffda97be08fc23f33b...	2008.0	220000	other	
205663	husqvawgb	7f7292ffad724ebbe9ca860f515245368d714c84705b42...	2017.0	500000	fullstack engineer	
205664	vwwgrxnt	cb25cc7304e9a24facda7f5567c7922ffc48e3d5d6018c...	2021.0	700000	qa engineer	
205665	zgn vuurxwvmrt	fb46a1a2752f5f652ce634f6178d0578ef6995ee59f6c8...	2019.0	5100000	frontend engineer	
205666	bgqsvz onvzrtj	0bcfc1d05f2e8dc4147743a1313aa70a119b41b30d4a1f...	2014.0	1240000	associate software developer	

205667 rows × 11 columns

In [52]:
```
df_merged3.drop('mean_ctc', axis=1, inplace=True)
df_merged3
```

Out[52]:

	company_hash	email_hash	orgyear	ctc	job_position	ctc_upda
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	1100000	other	
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018.0	449999	fullstack engineer	
2	Others	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2015.0	2000000	backend engineer	
3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	2017.0	700000	backend engineer	
4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	2017.0	1400000	fullstack engineer	
...	...	...	...	...	...	
205662	vuurt xzw	70027b728c8ee901fe979533ed94ffda97be08fc23f33b...	2008.0	220000	other	
205663	husqvawgb	7f7292ffad724ebbe9ca860f515245368d714c84705b42...	2017.0	500000	fullstack engineer	
205664	vwwgrxnt	cb25cc7304e9a24facda7f5567c7922ffc48e3d5d6018c...	2021.0	700000	qa engineer	
205665	zgn vuurxwvmrt	fb46a1a2752f5f652ce634f6178d0578ef6995ee59f6c8...	2019.0	5100000	frontend engineer	
205666	bgqsvz onvzrtj	0bcfc1d05f2e8dc4147743a1313aa70a119b41b30d4a1f...	2014.0	1240000	associate software developer	

205667 rows × 10 columns

Exploratory Data Analysis

In [54]:
```
#Top 10 employees earning more than most of the employees in the company.

df_merged3.sort_values(by='Tier', ascending=True)[:11]
```

Out[54]:

	company_hash	email_hash	orgyear	ctc	job_position	ctc_upda
102833	gdvzt uvqnztqo rxbxnta	6bf868fbad373e13d7e1101bdf8b2e7574c601ac58da22...	2013.0	1420000	fullstack engineer	
156864	fvrbvqn rvmo	f482a52c5062e9afcd19281b2a9193b9a6d8753d471102...	2011.0	5100000	backend engineer	
59297	wxmtq srgmvr	7c0a12af8ef03e226c7410750c466eaf38f426d347a90f...	2005.0	1500000	product designer	
59308	uvjbtznho	6f78cbafec28fef2de3ed990de0f6db5a6b213f4087c08...	2020.0	850000	fullstack engineer	
59311	vbvatho xn sqghu	eb78866d0ea55afc2f8fbd701d8a506503e6bbbdf0e134...	2013.0	1800000	backend engineer	
59315	wxnqxd	006632b251b7437440d48c7d47160fcaf00f370f2401a2...	2021.0	3600000	backend engineer	
156847	Others	f6b459ab0408279c9ece9fff1f7dd8f8e6a5ddb76a37ac...	2016.0	3400000	fullstack engineer	
59318	Others	4f333d33e0004abbb5ff748b357fd9d737ea6de23e68e8...	2011.0	3500000	engineering leadership	
59320	mvqwrvjo wgqugqvnt mvzpxzs	1250975f9df051d7f81127ebdbd10a89b470986d68f1d0...	2014.0	2000000	engineering leadership	
156842	atigat	8bc6e17f4727c5dddd7851e937d2fc6c9651629d3c40a2...	2015.0	1500000	backend engineer	
59325	xzeg tast xzaxv rxbxnta	bce0df9409cdfef503c38bd768385d7d977246da2a7868...	2020.0	2400000	product manager	

In [55]:
```
#Top 10 employees of data science in each company earning more than their peers - Class 1

d_filtered = df_merged3[(df_merged3['job_position'] == 'data scientist') & (df_merged3['Class'] == 1) & (df

d_filtered.groupby('company_hash', group_keys=False).apply(lambda x: x.nlargest(10, 'ctc')).reset_index(drc
```

Out[55]:

	company_hash	email_hash	orgyear	ctc	job_position	ctc_updat
0	Others	be8afd4a4e5cedb6bfbabbc88f15e756d69c3f2491d02a...	2021.0	200000000	data scientist	
1	Others	72ed7ced98573f71c8f95bc8b75aac4f0677e8872c6bec...	2019.0	199800000	data scientist	
2	Others	ee8dd42d6ea8365909147d861c7978d19f727a8075ba96...	2020.0	102500000	data scientist	
3	Others	e7722fb701c61e5cad82c39ee8bf3debe160d429b72c64...	2015.0	100000000	data scientist	
4	Others	4ddef8762b7585c6ee7b8c06834778f3aa00eb3be312b0...	2020.0	100000000	data scientist	
...	...	...	...	...	...	
930	zxwt vwnxbxkt	41b39db3bc359c98a43825f654f6b4fa8680dc99207616...	2019.0	3900000	data scientist	
931	zxxn ntwyzgrgsxto	56bc8f0eb4db04e459bf41f51c12b3bbb9c3595d8a172d...	2007.0	5500000	data scientist	
932	zxzlvwvqn	2937acfa6802f83ff11ddbd3de1997b686107dad0c2b5d...	2019.0	1900000	data scientist	
933	zxzlvwvqn	2937acfa6802f83ff11ddbd3de1997b686107dad0c2b5d...	2019.0	1900000	data scientist	
934	zxztrtvuo	1cd0a52ed52dae24d605d9cdc8536499c10ce62bfb070f...	2014.0	2250000	data scientist	

935 rows × 10 columns

In [56]:
```
#Bottom 10 employees of data science in each company earning less than their peers - Class 3

d_filtered = df_merged3[(df_merged3['job_position'] == 'data scientist') & (df_merged3['Class'] == 3) & (df

d_filtered.groupby('company_hash', group_keys=False).apply(lambda x: x.nsmallest(10, 'ctc')).reset_index(dr
```

Out[56]:

	company_hash	email_hash	orgyear	ctc	job_position	ctc_updated
0	amo mvzp	fdca92d09d4bb96383be5cd8438d83578e6fa7d09cbee6...	2012.0	2000000	data scientist	2
1	amo mvzp	cb28bd98e365ca53860af89cd3c96df77d4dc41289899b...	2017.0	2100000	data scientist	2
2	amo mvzp	0177202b15c66938ea8a061468928e036a7af0d0c1e882...	2017.0	2100000	data scientist	2
3	amo mvzp	13ed1c77a6961a9683ad363845c62186d052dc0d1f48bc...	2003.0	2100000	data scientist	2
4	aqag mvzsvrgqt	dd529581b56bafc7f54cc7236e5f0c6ab793cddcae2618...	2018.0	1300000	data scientist	2
...	...	...	...	...	...	
317	zxzlvwvqn	e82cbd0e93c2c0bd0cdc818447ab3dde19b577598d8e9b...	2011.0	1820000	data scientist	
318	zxztrtvuo	f678c67bee8cad9370f6aaf4f4cc22ffd417fd753663c6...	2019.0	1250000	data scientist	2
319	zxztrtvuo	f678c67bee8cad9370f6aaf4f4cc22ffd417fd753663c6...	2019.0	1250000	data scientist	2
320	zxztrtvuo	3027ca561b65f99da2f65bf3d85c6bb5d5687c67e69e89...	2018.0	1370000	data scientist	2
321	zxztrtvuo	10d566c5fca40ffe1d133b79594d071880711ef480da9f...	2017.0	1400000	data scientist	2

322 rows × 10 columns

In [57]:
```
#Bottom 10 employees (earning less than most of the employees in the company)- Tier 3

df_merged3[df_merged3['Tier'] == 3].sort_values(by='ctc', ascending=True)[:10]
```

Out[57]:

	company_hash	email_hash	orgyear	ctc	job_position	ctc_updated_
135356	xzntqcxtfmxn	3505b02549ebe2c95840ac6f0a35561a3b4cbe4b79cdb1...	2014.0	2	backend engineer	2
118179	xzntqcxtfmxn	f2b58aeed3c074652de2cfd3c0717a5d21d6fbcf342a78...	2013.0	6	some random title	2
114110	xzntqcxtfmxn	23ad96d6b6f1ecf554a52f6e9b61677c7d73d8a409a143...	2013.0	14	some random title	2
184805	Others	b8a0bb340583936b5a7923947e9aec21add5ebc50cd60b...	2016.0	15	other	2
183664	Others	75357254a31f133e2d3870057922feddeba82b88056a07...	2019.0	16	other	2
54808	Others	8786759b95d673466e94f62f1b15e4f8c6bd7de6164074...	2020.0	24	other	2
91523	Others	512f761579fb116e215cabc9821c7f81153f0763e16018...	2016.0	25	android engineer	2
116891	Others	f7e5e788676100d7c4146740ada9e2f8974defc01f571d...	2022.0	200	other	2
166280	Others	c411a6917058b50f44d7c62751be9b232155b23211de4c...	2013.0	300	database administrator	2
82002	Others	edcfb902656b736e1f35863298706d9d34ee795b7ed85a...	2018.0	500	cofounder	2

In [58]:
```
#Top 10 companies (based on their CTC)

df_merged3.groupby('company_hash')['ctc'].mean().reset_index().sort_values(by='ctc', ascending=False)[:10]
```

Out[58]:

	company_hash	ctc
3347	xzaxvmhrro	5.374143e+07
1421	obvqnuqxdwgb	4.395259e+07
798	ho tzsxzttqxzs wgbuvzj	4.348714e+07
1629	ouxwtltn rxbxnta	4.058400e+07
1699	ovxzn sgmvxz srvoo xzaxv ucnrna	4.056800e+07
113	axowgctq agrrvq	4.054400e+07
615	exqonoghqwt	4.038480e+07
2527	uyxrxuo xzaxv	3.770400e+07
639	fgqraihvzn rrw	3.495833e+07
513	egdwgzz	3.423250e+07

In [59]:
```
#Top 2 positions in every company (based on their CTC)

df_merged3.groupby(['company_hash', 'job_position'])['ctc'].mean().reset_index().sort_values(['company_hash
```

Out[59]:

	company_hash	job_position	ctc
64	Others	data entry	1.000000e+08
266	Others	telar	1.000000e+08
281	a ntwyzgrgsxto	other	3.716667e+06
278	a ntwyzgrgsxto	backend engineer	9.000000e+05
285	aaqxctz avnv owxtzwto vzvrjnxwo ucn rna	other	3.600000e+06
...	...	...	...
22707	zxyxrtzn ntwyzgrgsxto	backend engineer	9.925000e+05
22719	zxzlvwvqn	product manager	4.900000e+06
22717	zxzlvwvqn	fullstack engineer	2.181250e+06
22721	zxztrtvuo	backend architect	3.750000e+06
22724	zxztrtvuo	devops engineer	2.700000e+06

7437 rows × 3 columns

In [60]:
```
#Top 10 employees in each company - X department - having 5/6/7 years of experience earning more than their

df_filered1 = df_merged3[((df_merged3['years_of_experience'] == 5) | (df_merged3['years_of_experience'] ==

df_filered1.groupby('company_hash')['ctc'].mean().reset_index().sort_values(by='ctc', ascending=False).grou
```

Out[60]:

	company_hash	ctc
750	obvqnuqxdwgb	255555555.0
977	qtwpgzojo ntwy rvmo ucn rna	200000000.0
877	ovxzn sgmvxz srvoo xzaxv ucnrna	200000000.0
635	nqvctrnqxvzsrt	200000000.0
668	ntvwy egq xzaxv	200000000.0
...	...	...
451	jvzatd	100000.0
1168	tuxw	90000.0
804	onvnt evqb	75000.0
308	ftmvrg	70000.0
1684	xzaxvzv hzxctqoxnj mrggbxzsngz	66000.0

1911 rows × 2 columns

In [61]:
```
df_final = df_merged3.copy()
```

In [62]:
```
df_final
```

Out[62]:

	company_hash	email_hash	orgyear	ctc	job_position	ctc_upda
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	1100000	other	
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018.0	449999	fullstack engineer	
2	Others	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2015.0	2000000	backend engineer	
3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	2017.0	700000	backend engineer	
4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	2017.0	1400000	fullstack engineer	
...	...	...	...	...	...	
205662	vuurt xzw	70027b728c8ee901fe979533ed94ffda97be08fc23f33b...	2008.0	220000	other	
205663	husqvawgb	7f7292ffad724ebbe9ca860f515245368d714c84705b42...	2017.0	500000	fullstack engineer	
205664	vwwgrxnt	cb25cc7304e9a24facda7f5567c7922ffc48e3d5d6018c...	2021.0	700000	qa engineer	
205665	zgn vuurxwvmrt	fb46a1a2752f5f652ce634f6178d0578ef6995ee59f6c8...	2019.0	5100000	frontend engineer	
205666	bgqsvz onvzrtj	0bcfc1d05f2e8dc4147743a1313aa70a119b41b30d4a1f...	2014.0	1240000	associate software developer	

205667 rows × 10 columns

Handling Outliers.

In [64]:
```
#Dropping samples with unrealistic years of experience.

df_final = df_final[~(df_final['years_of_experience'] > 60)]
```

In [65]:
```
#Clipping 'ctc' column with values at 1st and 99th percentile.

lower = np.percentile(df_final['ctc'], 1)
upper = np.percentile(df_final['ctc'], 99)

df_final['ctc'] = df_final['ctc'].clip(lower=lower, upper=upper)
```

In [66]:
```
df_final
```

Out[66]:

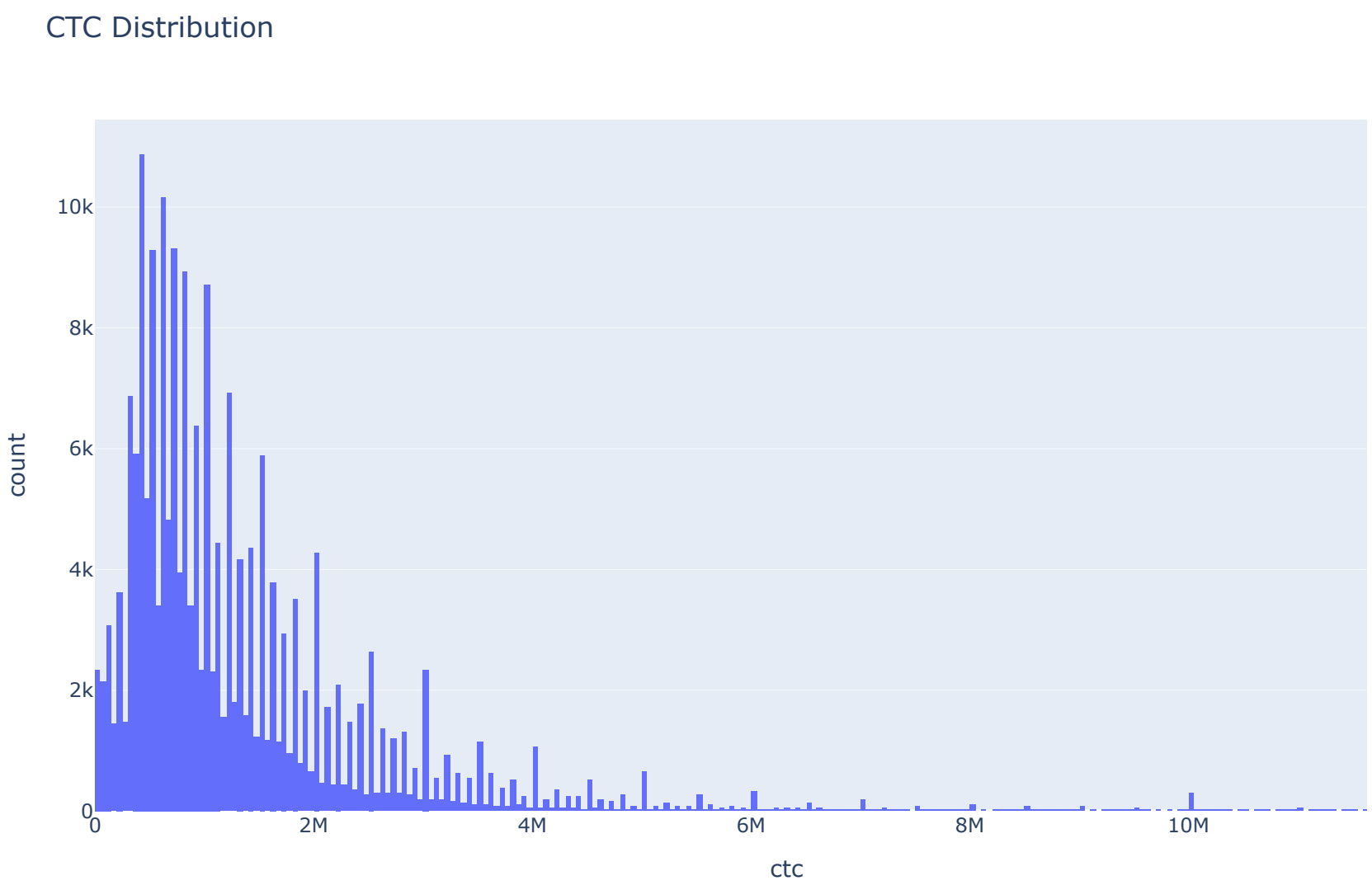
	company_hash	email_hash	orgyear	ctc	job_position	ctc_upda
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	1100000	other	
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018.0	449999	fullstack engineer	
2	Others	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2015.0	2000000	backend engineer	
3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	2017.0	700000	backend engineer	
4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	2017.0	1400000	fullstack engineer	
...	...	...	...	...	...	
205662	vuurt xzw	70027b728c8ee901fe979533ed94ffda97be08fc23f33b...	2008.0	220000	other	
205663	husqvawgb	7f7292ffad724ebbe9ca860f515245368d714c84705b42...	2017.0	500000	fullstack engineer	
205664	vwwgrxnt	cb25cc7304e9a24facda7f5567c7922ffc48e3d5d6018c...	2021.0	700000	qa engineer	
205665	zgn vuurxwvmrt	fb46a1a2752f5f652ce634f6178d0578ef6995ee59f6c8...	2019.0	5100000	frontend engineer	
205666	bgqsvz onvzrtj	0bcfc1d05f2e8dc4147743a1313aa70a119b41b30d4a1f...	2014.0	1240000	associate software developer	

205623 rows × 10 columns

In [67]:
```
#Clipping 'orgyear' column with values at 1st and 99th percentile.

lower = np.percentile(df_final['orgyear'], 1)
upper = np.percentile(df_final['orgyear'], 99)
```

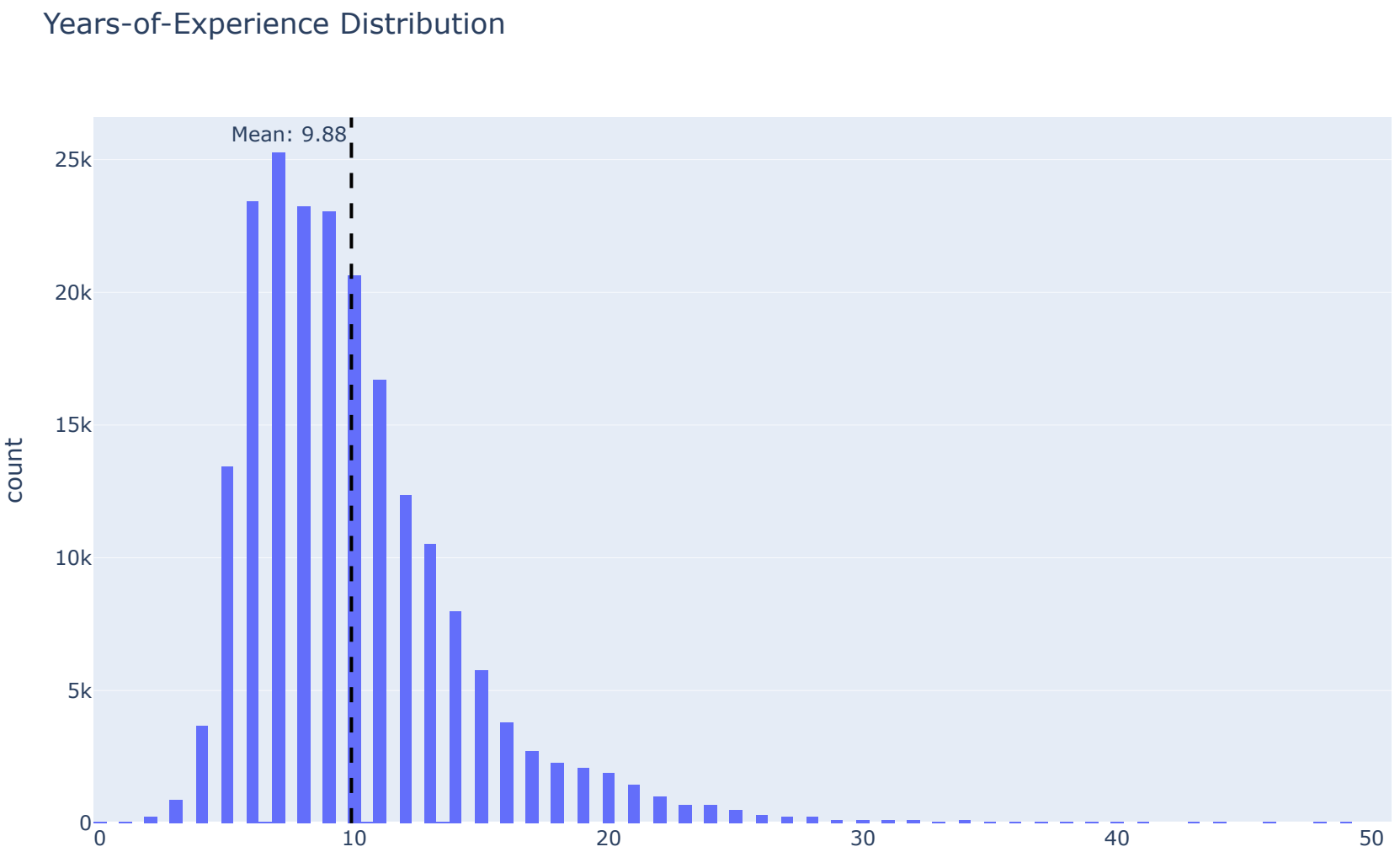
```
df_final['orgyear'] = df_final['orgyear'].clip(lower=lower, upper=upper)
```

In [68]:
```
#Distribution of 'ctc'.

px.histogram(df_final, x='ctc', title='CTC Distribution', width=1000, height=600)
```



We can observe a strongly right skewed normal distribution of income, indicating outliers.

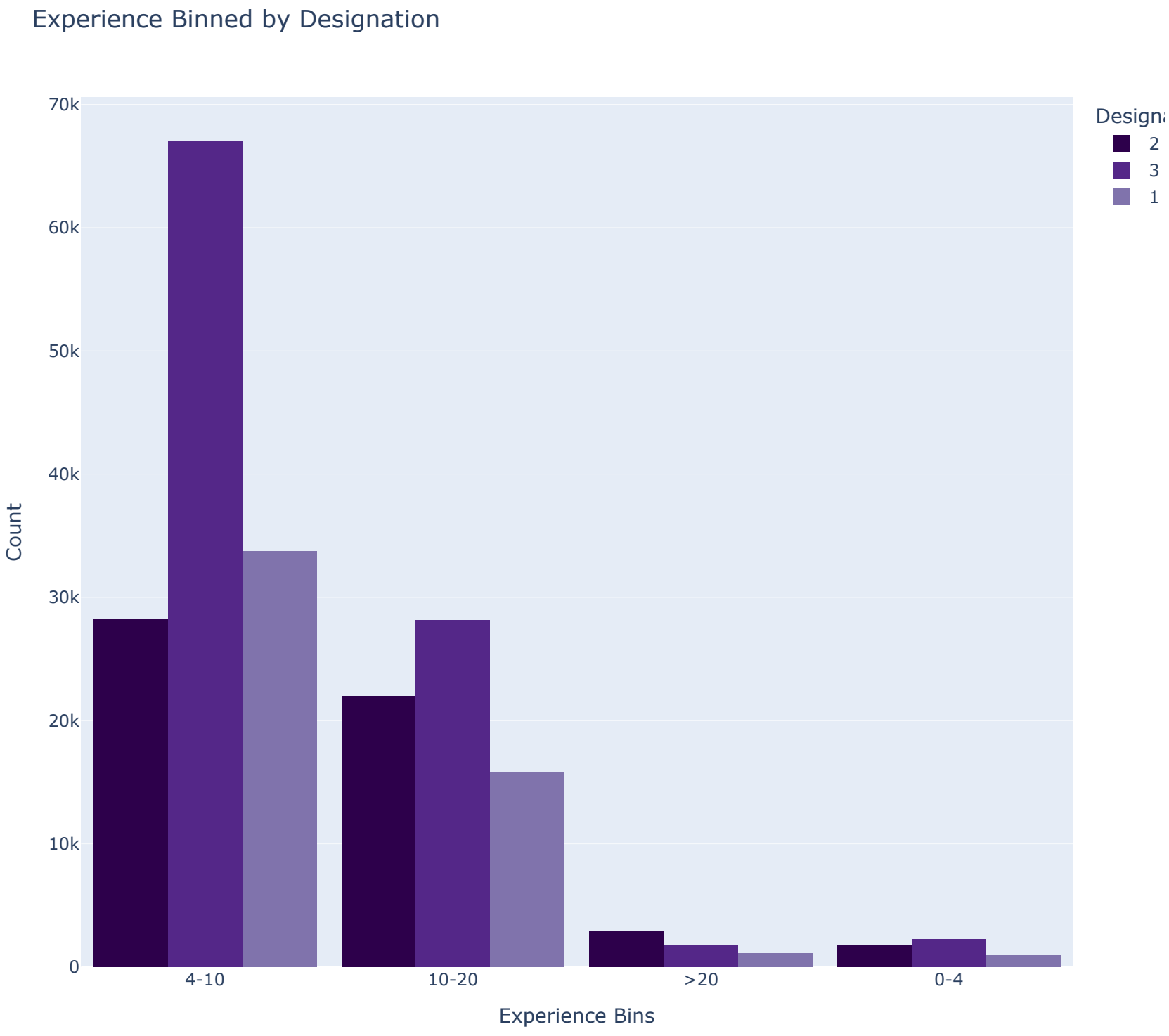
In [70]:
```
#Distribution of 'years_of_experience'.

fig1 = px.histogram(df_final, x='years_of_experience', title='Years-of-Experience Distribution', width=1000

mean_val = df_final['years_of_experience'].mean()
median_val = df_final['years_of_experience'].median()

fig1.add_vline(x=mean_val, line_dash="dash", line_color="black", annotation_text=f"Mean: {mean_val:.2f}", a
```



Slightly right skewed distribution of years of experience.

In [72]:
```
bins = [0,4,10,20,float('inf')]
labels = ['0-4', '4-10', '10-20', '>20']
df_final['exp_bin'] = pd.cut(df_final['years_of_experience'],
                             labels=labels,
                             bins=bins)
```

In [73]:
```
fig = px.histogram(df_final,
                   x='exp_bin',
                   color='designation',
                   barmode='group',
                   title="Experience Binned by Designation",
                   color_discrete_sequence=px.colors.diverging.PuOr_r
                  )

fig.update_layout(
    width=900,
    height=800,
    xaxis_title="Experience Bins",
    yaxis_title="Count",
    legend_title="Designation",
    bargap=0.1
)

fig.show()
```

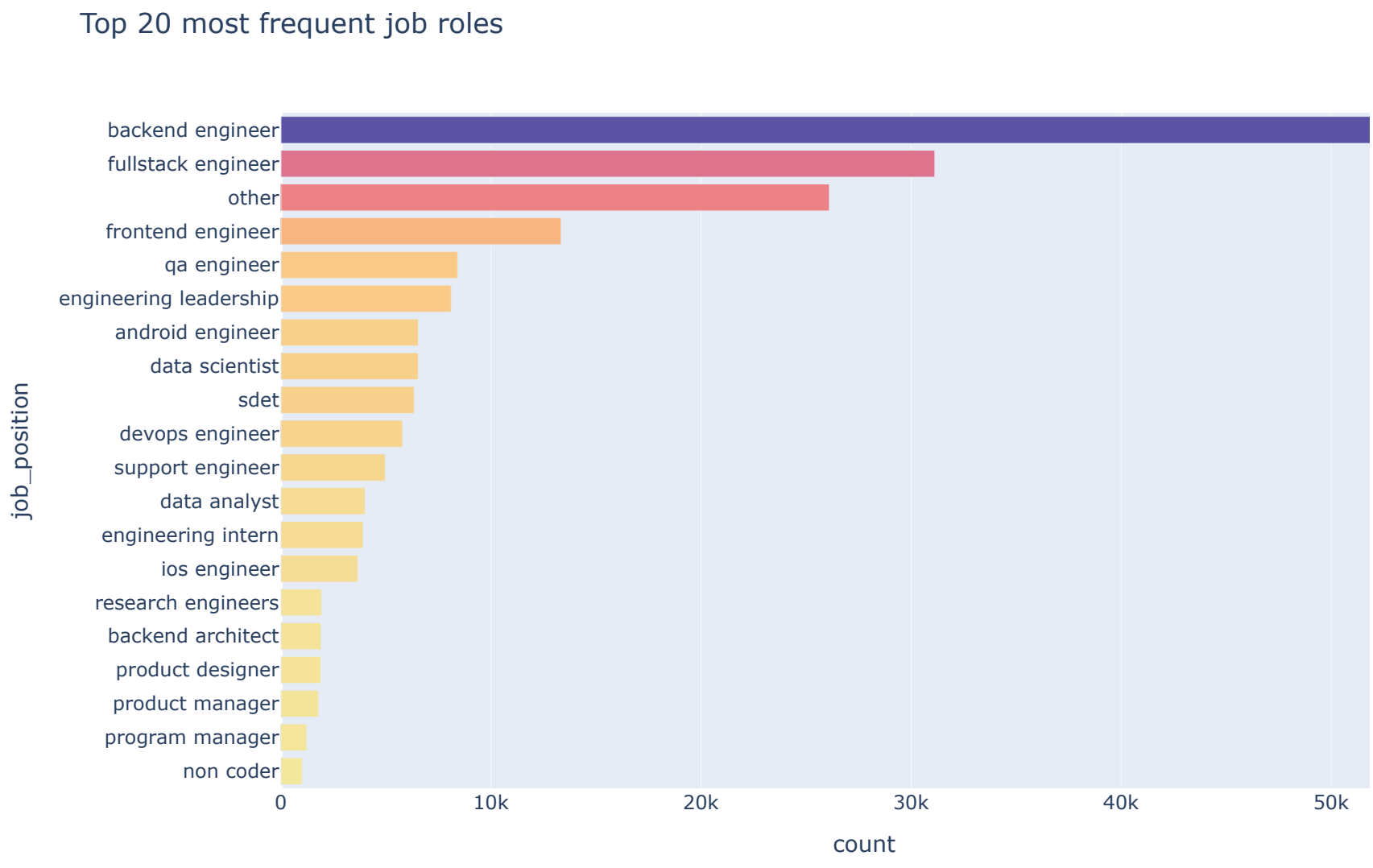


We can see a higher number of learners in the 4-10 years of range. We further notice a higher interest for designation 3 - which are people that earn less than average of their company's department having same years of experience. A similar trend can be seen for 10-20 years as well.

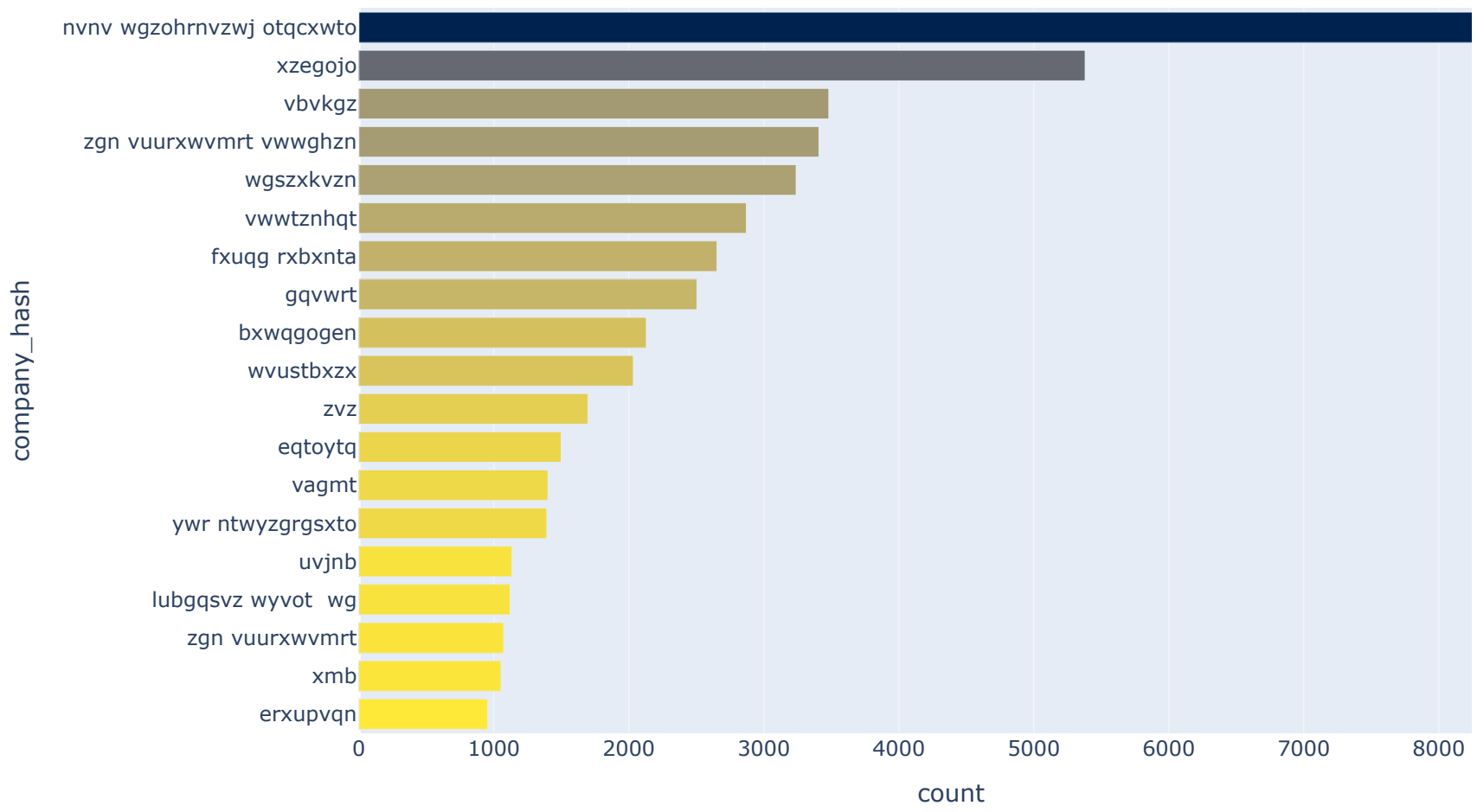
In [75]:
```
px.scatter(df_final, x='years_of_experience', y='ctc', width=1000, height=600, color='designation', title='
```

No discernible pattern seen between experience and ctc.

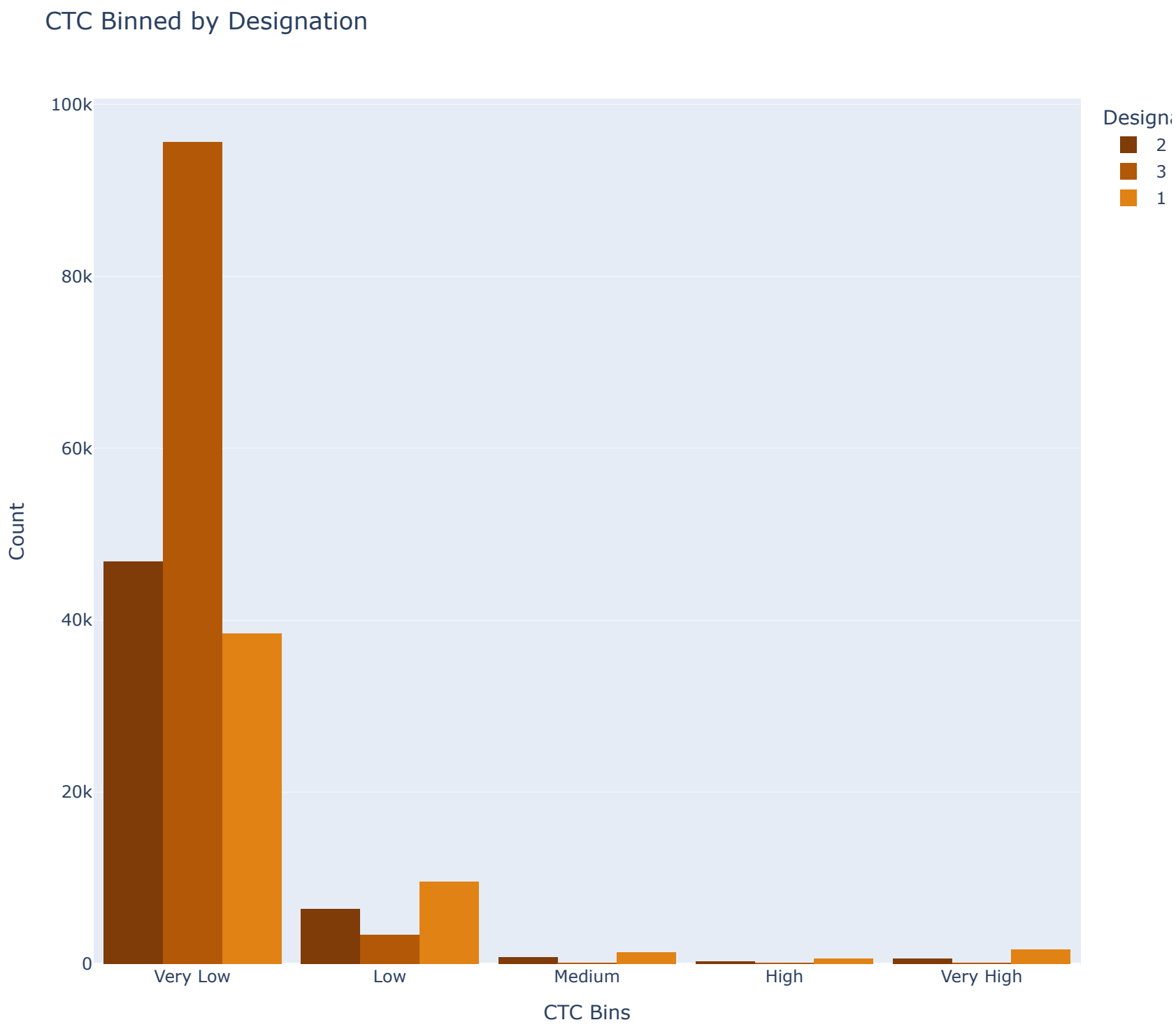
In [77]:
```
df_temp2 = df_final['job_position'].value_counts().reset_index().sort_values(by='count', ascending=True).ta

px.bar(df_temp2, x='count', y='job_position', title='Top 20 most frequent job roles', width=1000, height=60
```



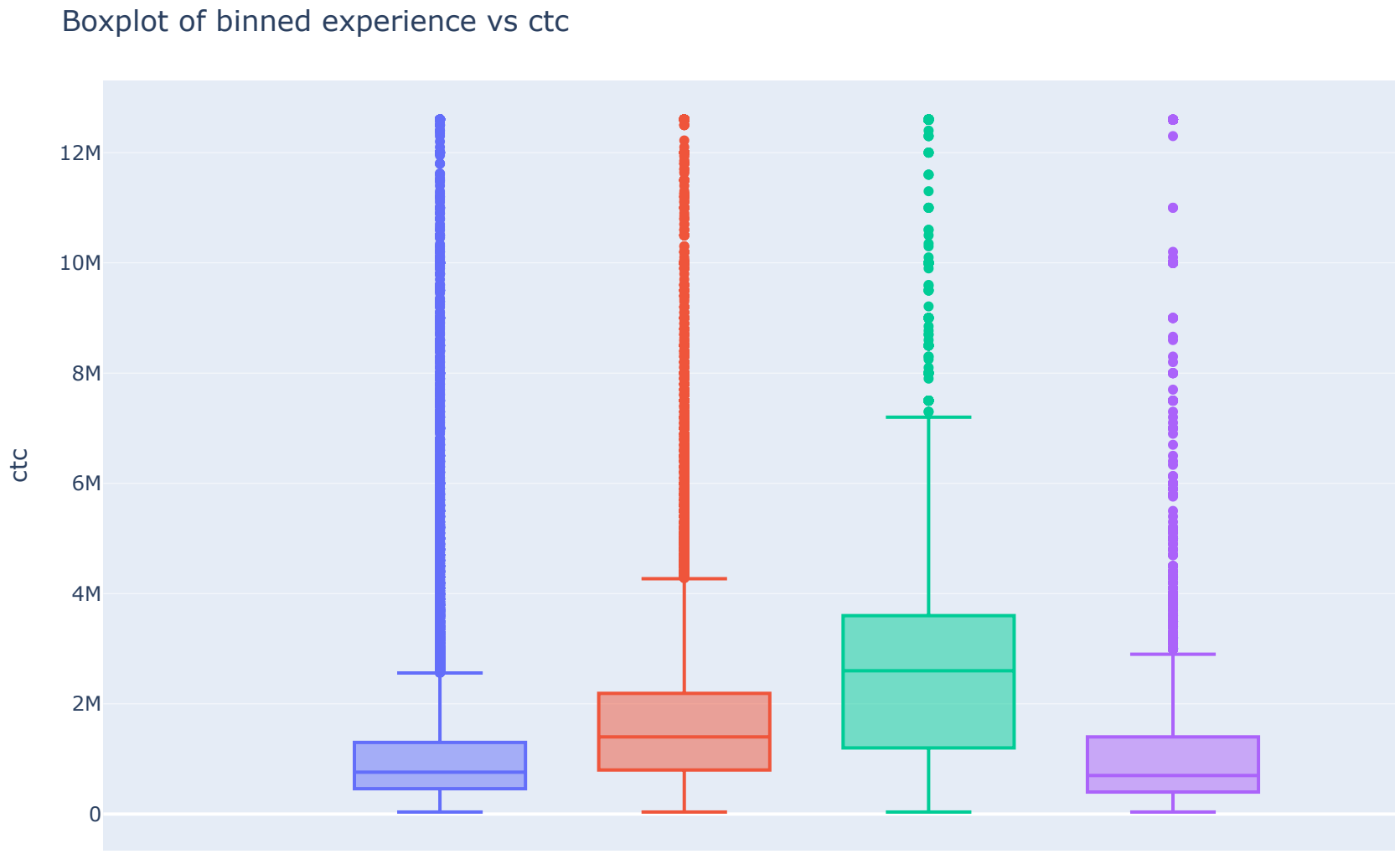
In [78]:
```
df_temp3 = df_final['company_hash'].value_counts().reset_index().sort_values(by='count', ascending=True).ta

px.bar(df_temp3[~(df_temp3['company_hash'] == 'Others')], x='count', y='company_hash', title='Top 20 most f
```



In [79]:
```
min_ctc = 37000
max_ctc = 12600000

bins = pd.cut(df_final['ctc'], bins=5, labels=['Very Low', 'Low', 'Medium', 'High', 'Very High'])
df_final['ctc_category'] = bins

fig = px.histogram(df_final,
                   x='ctc_category',
                   color='designation',
                   barmode='group',
                   title="CTC Binned by Designation",
                   color_discrete_sequence=px.colors.diverging.PuOr
                  )

fig.update_layout(
    width=900,
    height=800,
    xaxis_title="CTC Bins",
    yaxis_title="Count",
    legend_title="Designation",
    bargap=0.1
)

fig.show()
```

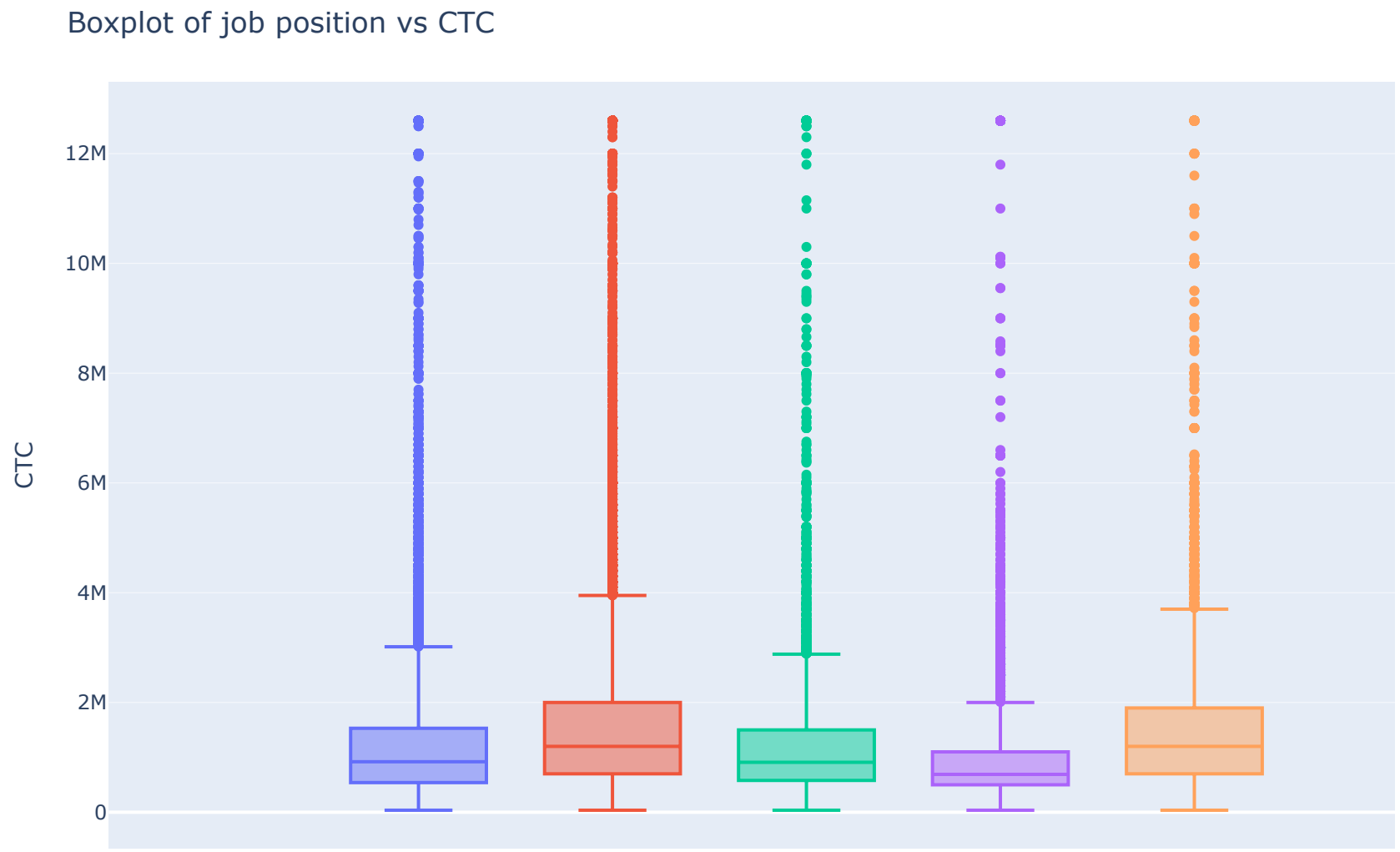


In [80]:
```
fig_4 = px.box(df_final, y="ctc", color = "exp_bin")

fig_4.update_layout(
    width=1100,
    height=600,
    xaxis_title="experience(in years)",
    yaxis_title="ctc",
    legend_title="years_of_experience",
    title='Boxplot of binned experience vs ctc'
)

fig_4.show()
```



We can observe a number of outliers for specified bins of experience years.

In [82]:
```
df_temp4 = df_final[(df_final['job_position'] == 'backend engineer') | (df_final['job_position'] == 'fullst
                    (df_final['job_position'] == 'andriod engineer') | (df_final['job_position'] == 'data s

fig_5 = px.box(df_temp4, y="ctc", color = "job_position")

fig_5.update_layout(
    width=1100,
    height=600,
    xaxis_title="Job Position",
    yaxis_title="CTC",
    legend_title="Job Position",
    title='Boxplot of job position vs CTC'
)

fig_5.show()
```



We can oberserve a variying range of salaries for the job positions of interest.

In [84]:
```
df_final
```

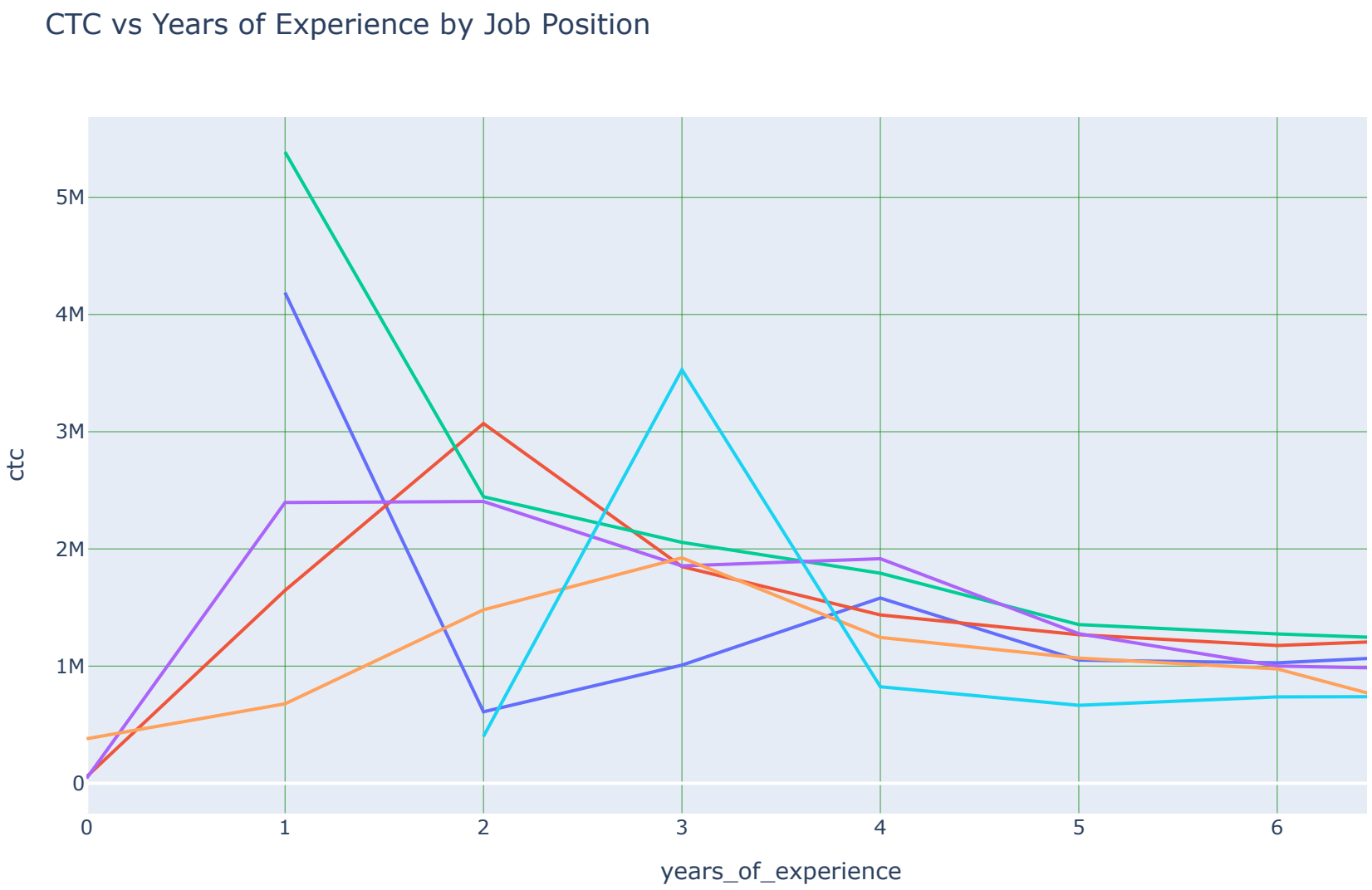
Out[84]:

	company_hash	email_hash	orgyear	ctc	job_position	ctc_upda
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	1100000	other	
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018.0	449999	fullstack engineer	
2	Others	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2015.0	2000000	backend engineer	
3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	2017.0	700000	backend engineer	
4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	2017.0	1400000	fullstack engineer	
...	...	...	...	...	...	
205662	vuurt xzw	70027b728c8ee901fe979533ed94ffda97be08fc23f33b...	2008.0	220000	other	
205663	husqvawgb	7f7292ffad724ebbe9ca860f515245368d714c84705b42...	2017.0	500000	fullstack engineer	
205664	vwwgrxnt	cb25cc7304e9a24facda7f5567c7922ffc48e3d5d6018c...	2021.0	700000	qa engineer	
205665	zgn vuurxwvmrt	fb46a1a2752f5f652ce634f6178d0578ef6995ee59f6c8...	2019.0	5100000	frontend engineer	
205666	bgqsvz onvzrtj	0bcfc1d05f2e8dc4147743a1313aa70a119b41b30d4a1f...	2014.0	1240000	associate software developer	

205623 rows × 12 columns

In [85]:
```
df_job_exp = df_final[((df_final["job_position"] == 'backend engineer') | (df_final["job_position"] == 'ful
df_job_exp
df_grouped = df_job_exp.groupby(['job_position','years_of_experience'])['ctc'].mean().reset_index()
df_grouped.head()
fig_6 = px.line(df_grouped,
                x="years_of_experience",
                y="ctc",
                color="job_position",
                title="CTC vs Years of Experience by Job Position")

fig_6.update_layout(
    width=1100,
    height=600,
```

```
    xaxis=dict(showgrid=True, gridcolor="green"),
    yaxis=dict(showgrid=True, gridcolor="green"),
)

fig_6.show()
```



Line Graph for the above job positions, comparing mean CTC with years of experience.

The graph exhibits anomalies, likely due to data quality issues.

Scaling

In [88]:
```
df_final
```

Out[88]:

	company_hash	email_hash	orgyear	ctc	job_position	ctc_upda
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	1100000	other	
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018.0	449999	fullstack engineer	
2	Others	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2015.0	2000000	backend engineer	
3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	2017.0	700000	backend engineer	
4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	2017.0	1400000	fullstack engineer	
...	...	...	...	...	...	
205662	vuurt xzw	70027b728c8ee901fe979533ed94ffda97be08fc23f33b...	2008.0	220000	other	
205663	husqvawgb	7f7292ffad724ebbe9ca860f515245368d714c84705b42...	2017.0	500000	fullstack engineer	
205664	vwwgrxnt	cb25cc7304e9a24facda7f5567c7922ffc48e3d5d6018c...	2021.0	700000	qa engineer	
205665	zgn vuurxwvmrt	fb46a1a2752f5f652ce634f6178d0578ef6995ee59f6c8...	2019.0	5100000	frontend engineer	
205666	bgqsvz onvzrtj	0bcfc1d05f2e8dc4147743a1313aa70a119b41b30d4a1f...	2014.0	1240000	associate software developer	

205623 rows × 12 columns

In [89]:
```
#Dropping unnecessary columns that do not contribute to clustering.

X = df_final[['ctc', 'years_of_experience', 'designation', 'Class', 'Tier']]
```

In [90]:
```
X
```

Out[90]:

	ctc	years_of_experience	designation	Class	Tier
0	1100000	9.0	2	1	3
1	449999	7.0	3	3	3
2	2000000	10.0	1	1	3
3	700000	8.0	3	3	3
4	1400000	8.0	2	1	1
...	...	...	...	...	...
205662	220000	17.0	2	3	3
205663	500000	8.0	2	3	3
205664	700000	4.0	1	1	3
205665	5100000	6.0	3	3	3
205666	1240000	11.0	1	1	3

205623 rows × 5 columns

In [91]:
```
from sklearn.preprocessing import StandardScaler
```

In [92]:
```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [93]:
```
X_scaled
```

Out[93]:
```
array([[-0.18800619, -0.20854201, -0.28101216, -1.50882673,  0.51528667],
       [-0.57816162, -0.68083909,  0.93204076,  0.71208733,  0.51528667],
       [ 0.3522082 ,  0.02760654, -1.49406507, -1.50882673,  0.51528667],
       ...,
       [-0.42810147, -1.38928471, -1.49406507, -1.50882673,  0.51528667],
       [ 2.21294666, -0.91698763,  0.93204076,  0.71208733,  0.51528667],
       [-0.10397284,  0.26375508, -1.49406507, -1.50882673,  0.51528667]])
```

Visualization using PCA and t-SNE

In [95]:
```
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import umap.umap_ as umap
```

In [96]:
```
#PCA with 2-Dimensions.

pca_2d = PCA(n_components=2)

X_embedded = pca_2d.fit_transform(X_scaled)

print(X_embedded.shape)
```

```
(205623, 2)
```

In [97]:
```
px.scatter(x=X_embedded[:,0], y=X_embedded[:,1], width=1000, height=600)
```

There are no clear clusters seen with PCA so we perform t-SNE.

In [99]:
```
#TSNE for 2 dimensions.

tsne = TSNE(perplexity=50)

X_embedded_tsne = tsne.fit_transform(X)

print(X_embedded_tsne.shape)
```

```
(205623, 2)
```

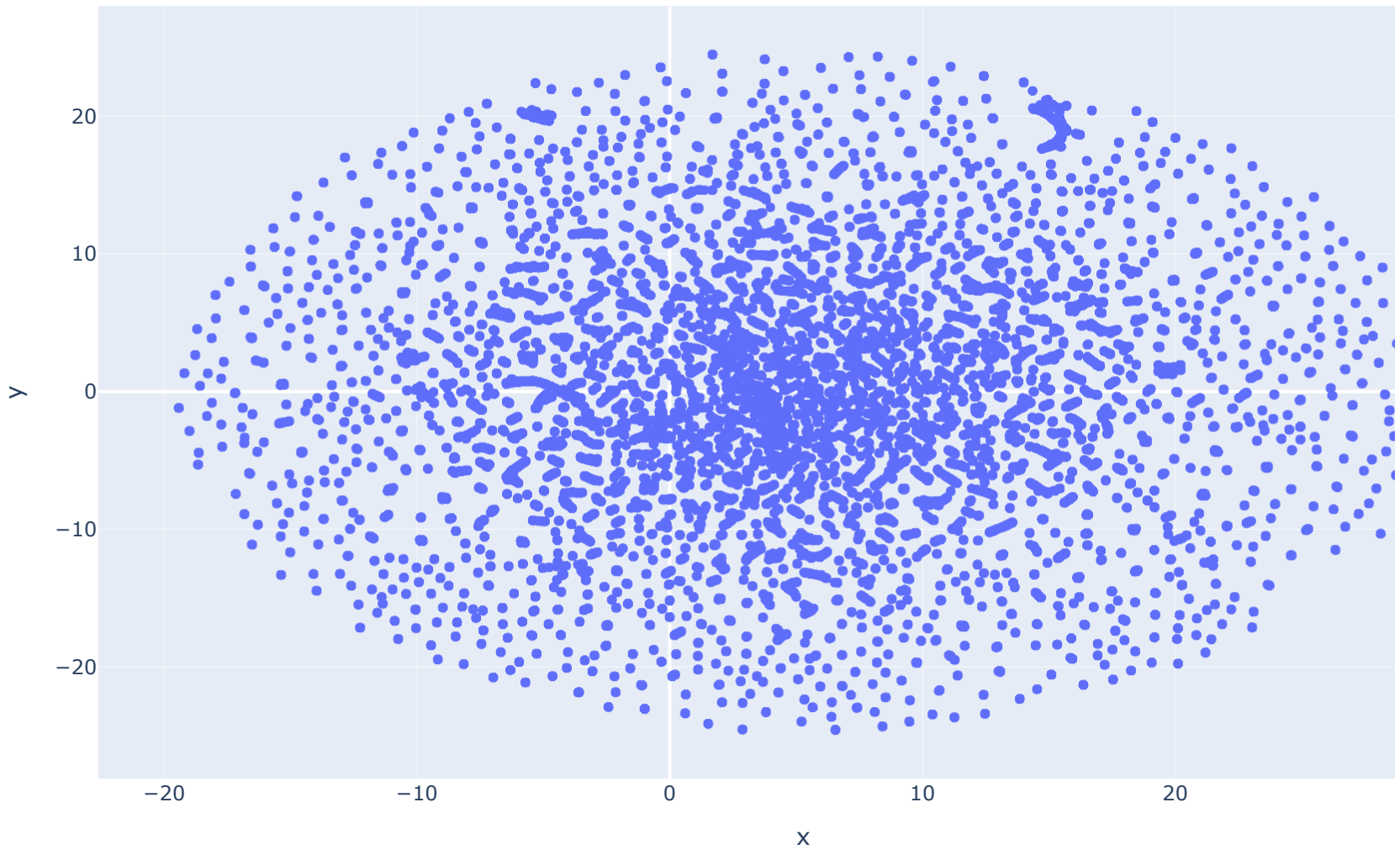
In [100…
```
px.scatter(x=X_embedded_tsne[:,0], y=X_embedded_tsne[:,1], width=1000, height=600)
```

Areas with varying densities can be observed but we still cannot make out on the possible number of clusters. There are some outliers on the edges.

In [102…
```
#UMAP for 2-dimensions

umap1 = umap.UMAP(n_components=2, random_state=42)

X_embedded_umap = umap1.fit_transform(X)

print(X_embedded_umap.shape)
```

```
(205623, 2)
```

In [103…
```
px.scatter(x=X_embedded_umap[:,0], y=X_embedded_umap[:,1], width=1000, height=600)
```

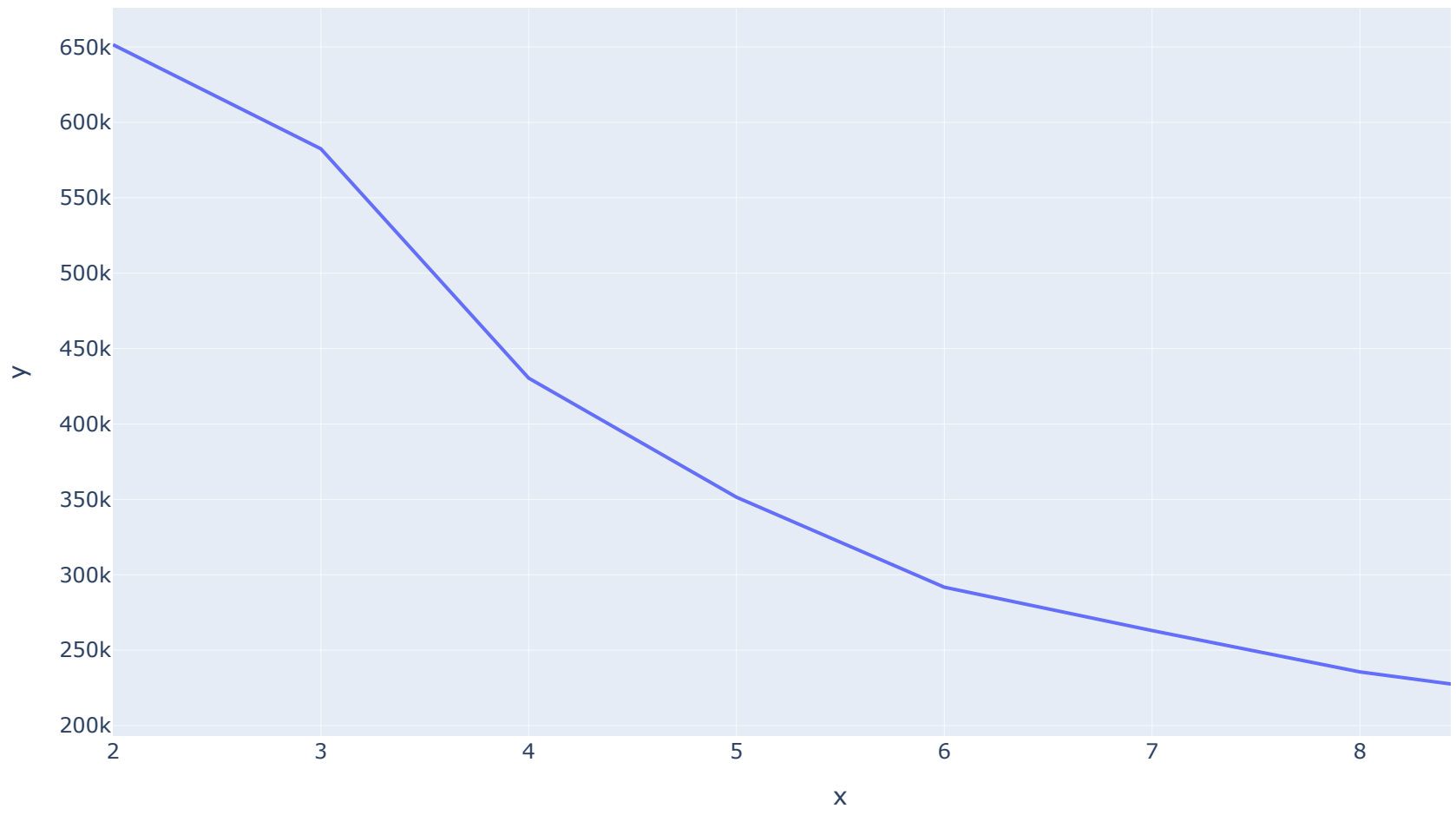


3 different densities can be observed, one at the center, one in the middle and one at the outer edges. This could indicate to 3 clusters.

k-Means Clustering

In [106…
```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

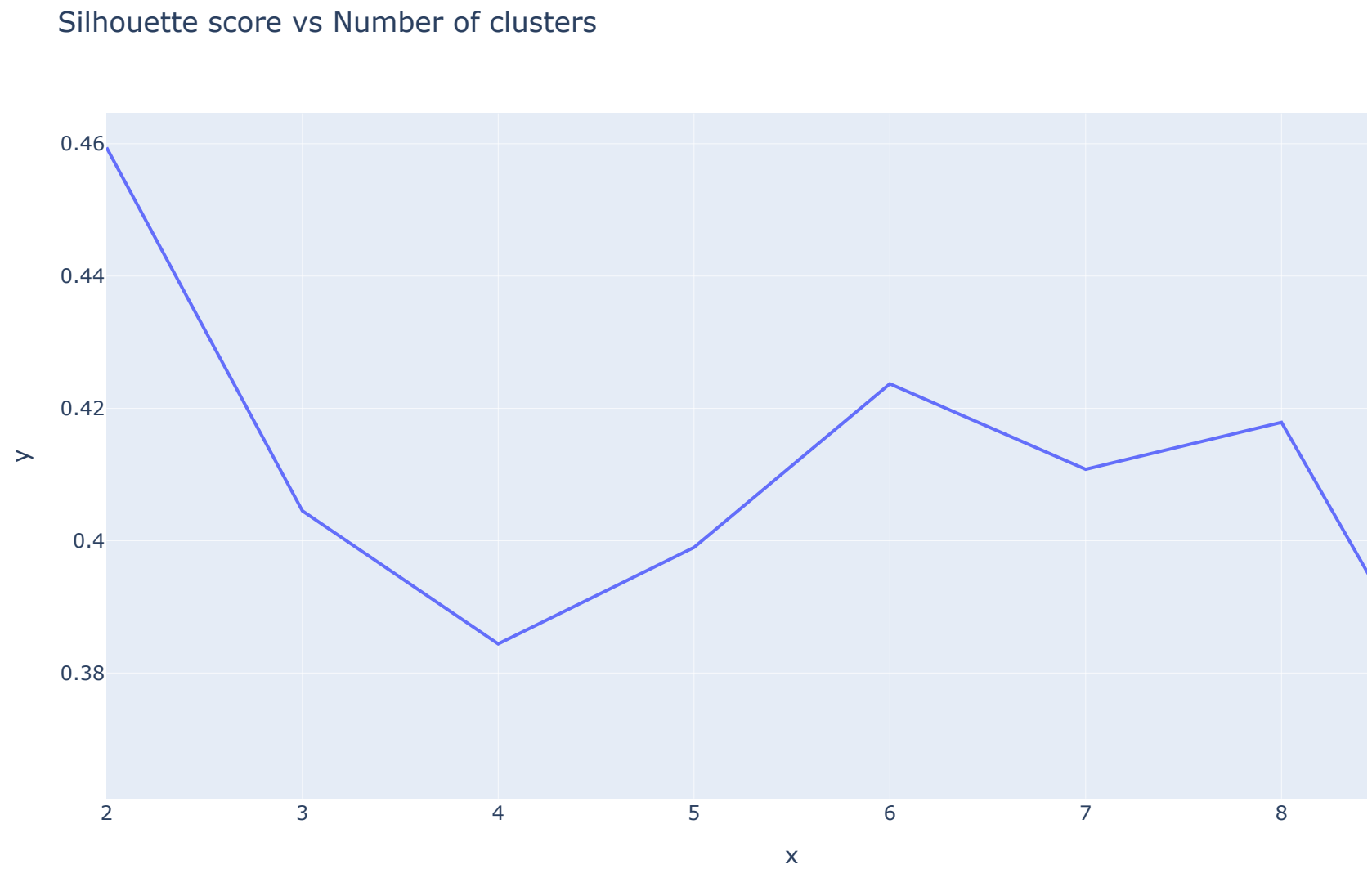
In [107…
```
#Using elbow method to find optimal 'k'.

elbow = []

for k in range(2,10):
    km_cluster = KMeans(n_clusters=k)
    km_cluster.fit(X_scaled)
    elbow.append(km_cluster.inertia_)

px.line(x=range(2,10), y=elbow, width=1000, height=600, title='WCSS vs Number of clusters')
```



We can see a smooth line, no sudden drop (or elbow) is observed for any value of 'k', so it is difficult to comment on the number of clusters. We now calculate Silhouette score for further analysis.

In [109…
```
#Finding Silhouette score for each 'k'.

silhouette = []

for k in range(2,10):
    km_cluster = KMeans(n_clusters=k)
    km_cluster.fit(X_scaled)
    silhouette.append(silhouette_score(X_scaled, km_cluster.labels_))
```

In [110…
```
px.line(x=range(2,10), y=silhouette, width=1000, height=600, title='Silhouette score vs Number of clusters'
```



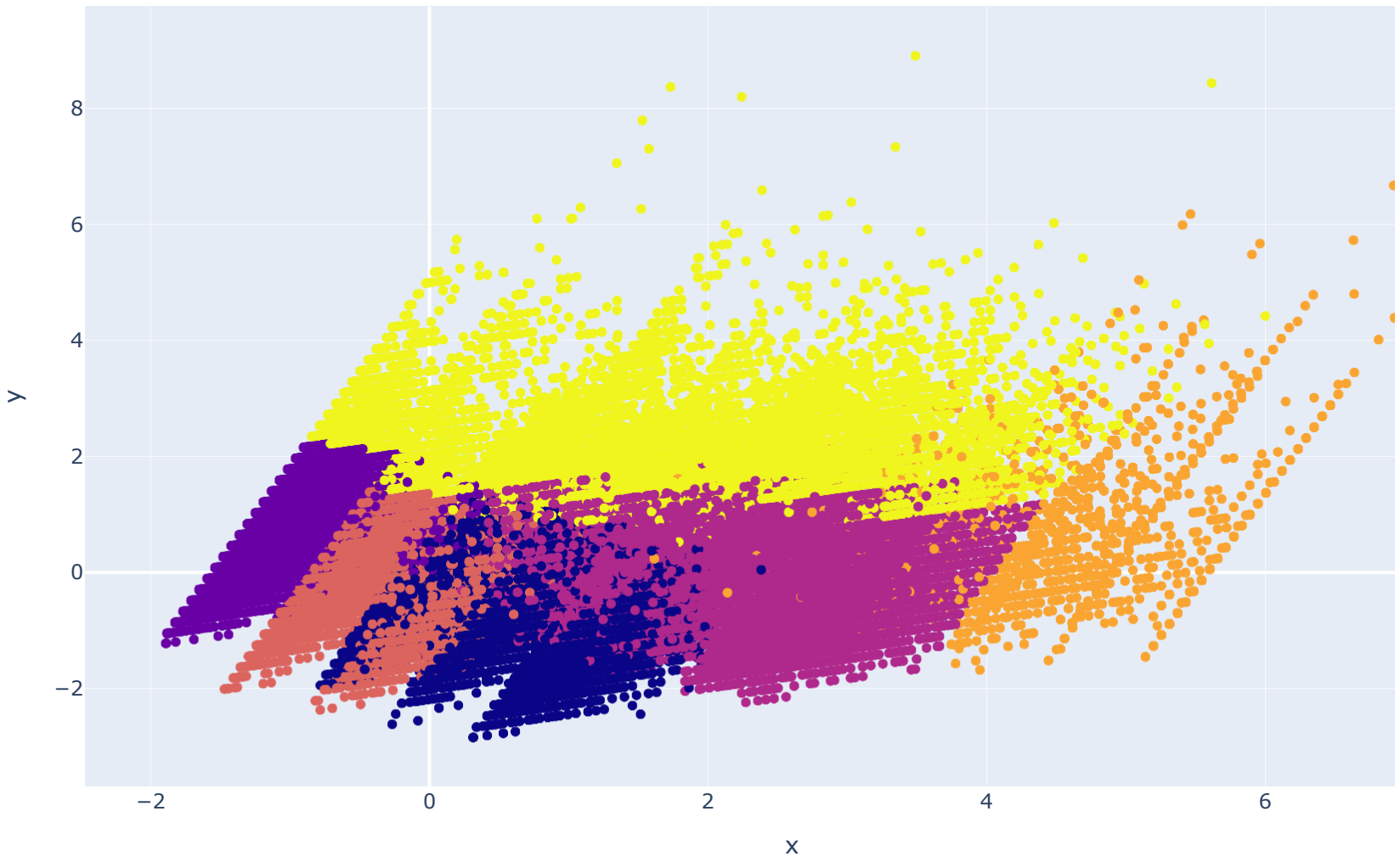
We get two peaks at k = 4 and 6. Comparing with the elbow graph, both seem to have nearly the same amount of drop in WCSS score.

Given the size of the dataset we decide 6 clusters for KMeans.

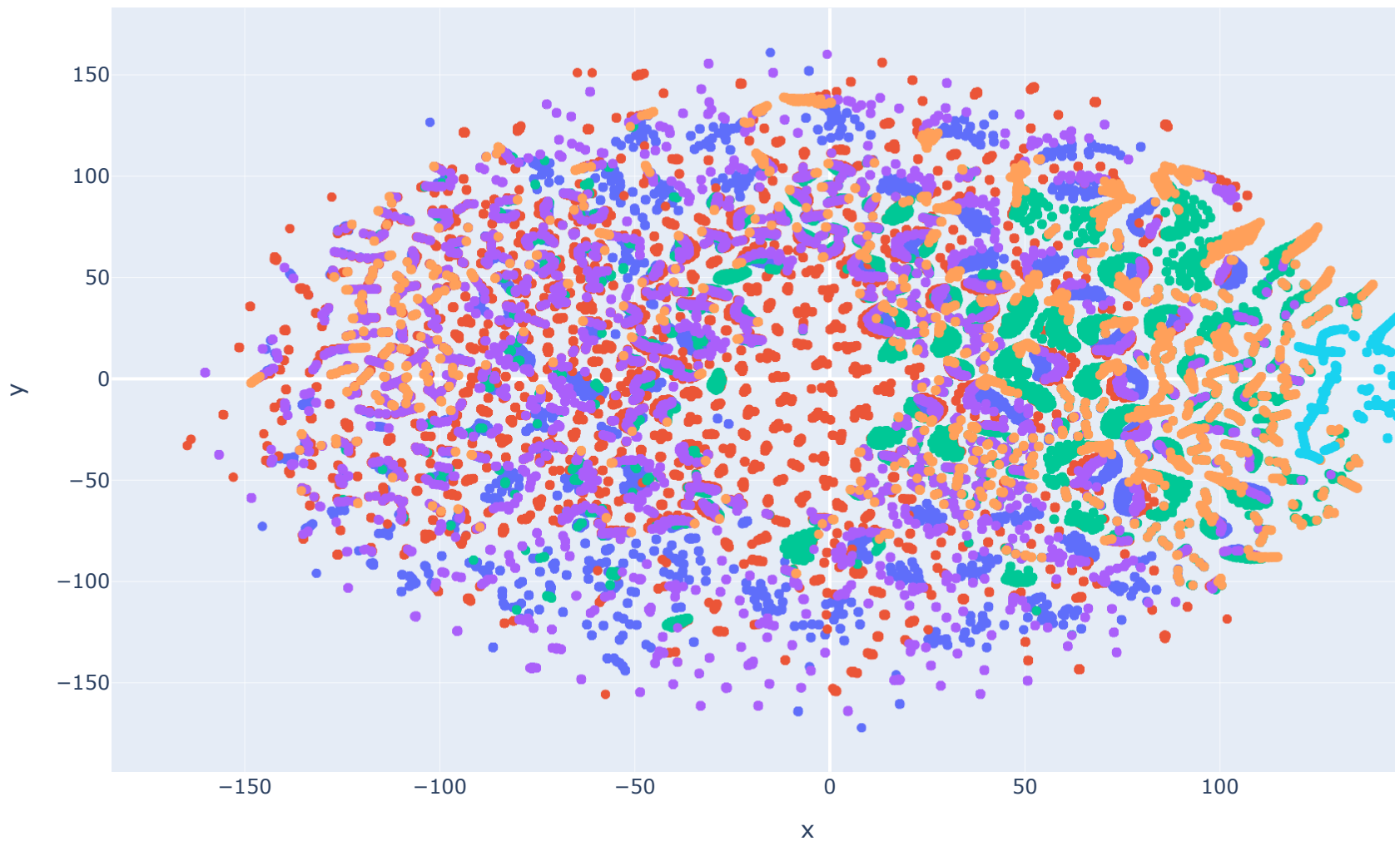
In [112…
```
# Clustering with optimal 'k'.

km_cluster_final = KMeans(n_clusters=6)
km_cluster_final.fit(X_scaled)
```

Out[112…
```
KMeans(n_clusters=6)
```

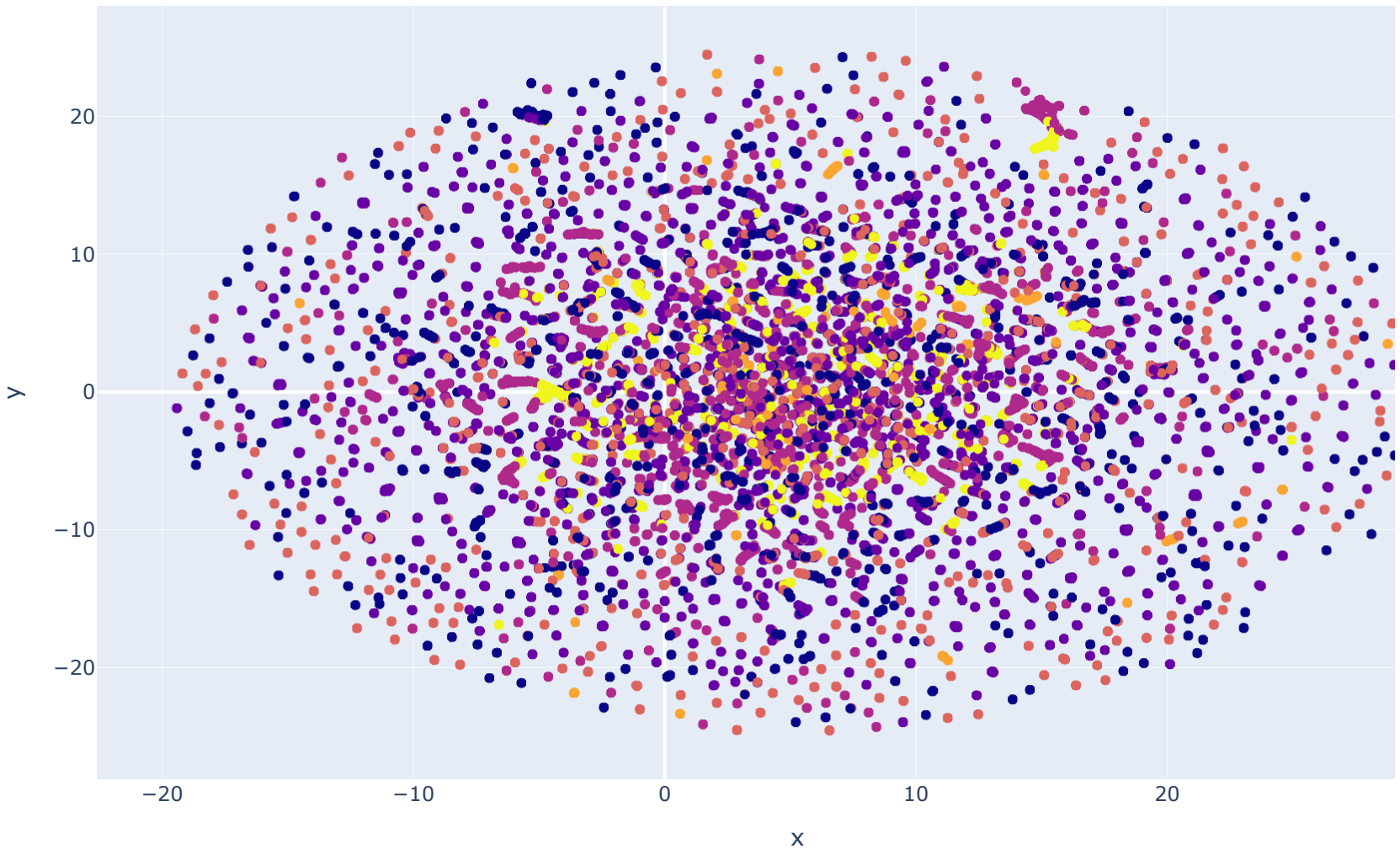
In [113…
```
#PCA visualization for KMeans clusters.

px.scatter(x=X_embedded[:,0], y=X_embedded[:,1], width=1000, height=600, color=km_cluster_final.labels_)
```



In [114…
```
#TSNE visualization for KMeans clusters.

px.scatter(x=X_embedded_tsne[:,0], y=X_embedded_tsne[:,1], width=1000, height=600, color=km_cluster_final.l
```



In [115…
```
px.scatter(x=X_embedded_umap[:,0], y=X_embedded_umap[:,1], width=1000, height=600, color=km_cluster_final.l
```



PCA and TSNE give some level of understanding of the clusters. In the TSNE visualization, we can see that similar shapes have been clustered together.

No discernible pattern can be seen in UMAP.

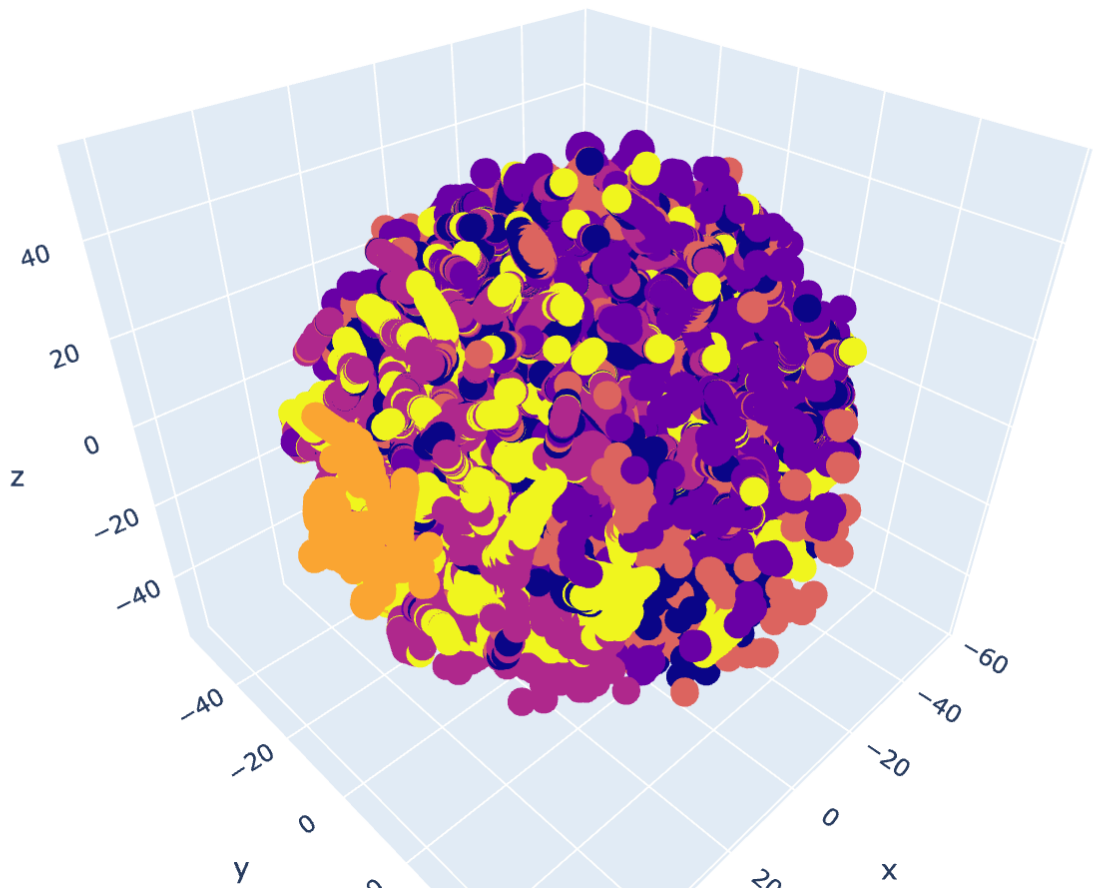
In [117…
```
#TSNE for 3 Dimensions.

tsne_3d = TSNE(n_components=3, perplexity=50)

X_embedded_tsne_3d = tsne_3d.fit_transform(X)

print(X_embedded_tsne_3d.shape)
```

```
(205623, 3)
```

In [118…
```
X_embedded_tsne_3d
```

Out[118…
```
array([[ -0.35373777, -28.138155  ,   3.565624  ],
       [-16.363283  ,  10.288775  , -15.858201  ],
       [ 24.420006  , -27.30653   ,  22.067749  ],
       ...,
       [-14.20627   , -44.648495  , -10.464795  ],
       [ 33.9332    , -25.340141  ,   1.111563  ],
       [  8.722746  ,  29.176739  ,   9.7907915 ]], dtype=float32)
```

In [119…
```
px.scatter_3d(x=X_embedded_tsne_3d[:,0], y=X_embedded_tsne_3d[:,1], z=X_embedded_tsne_3d[:,2], width=1000,
```



3-D representation of TSNE.

Clustering with Gaussian Mixture Models(GMM)

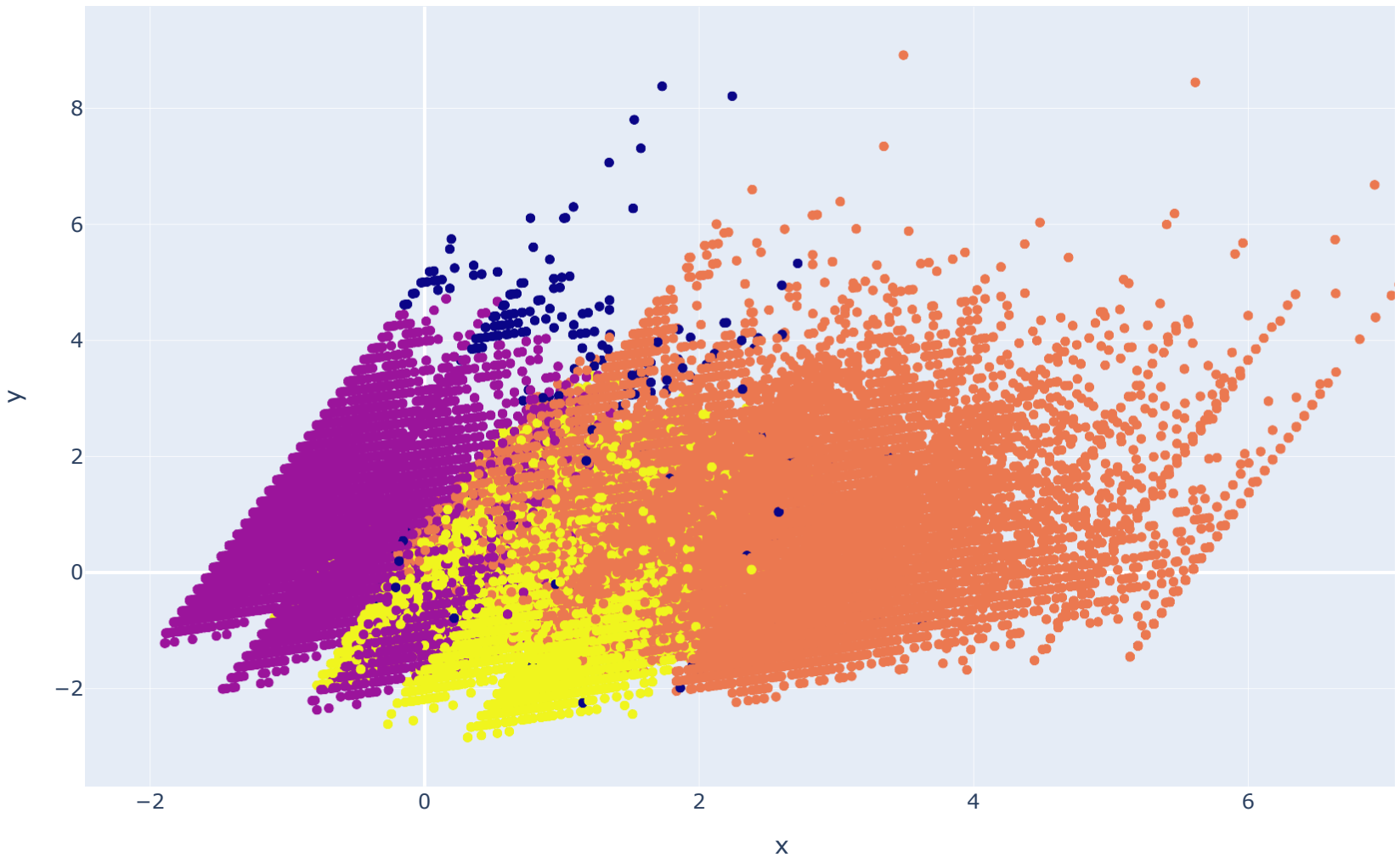
In [122…
```
from sklearn.mixture import GaussianMixture
```

In [123…
```
#GMM for 4 clusters.

gmm = GaussianMixture(n_components=4)
gmm.fit(X_scaled)
```

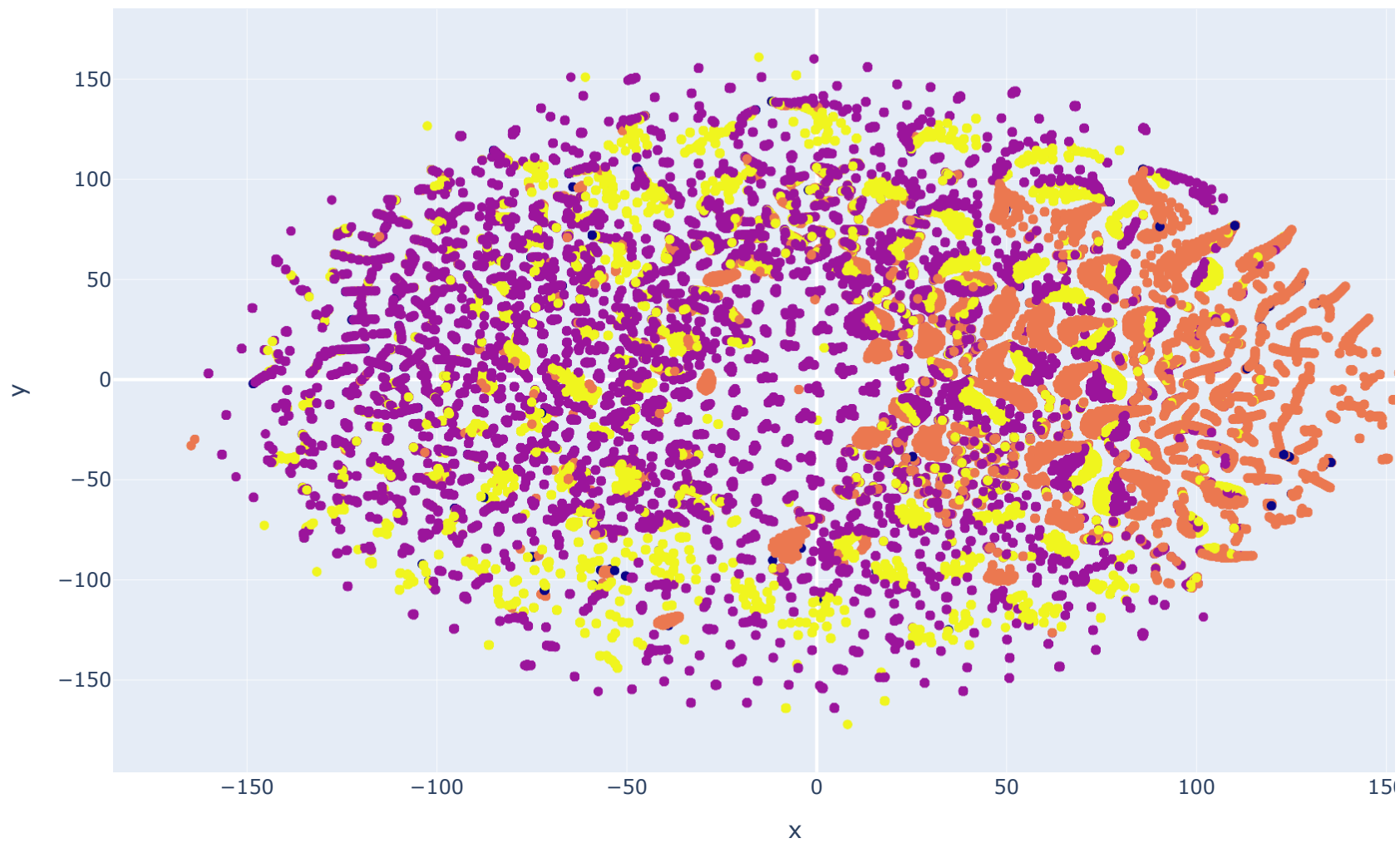
Out[123…
```
GaussianMixture(n_components=4)
```

In [124…
```
labels = gmm.predict(X_scaled)
```

In [125…
```
#PCA visualization for GMM.

px.scatter(x=X_embedded[:,0], y=X_embedded[:,1], width=1000, height=600, color=labels)
```

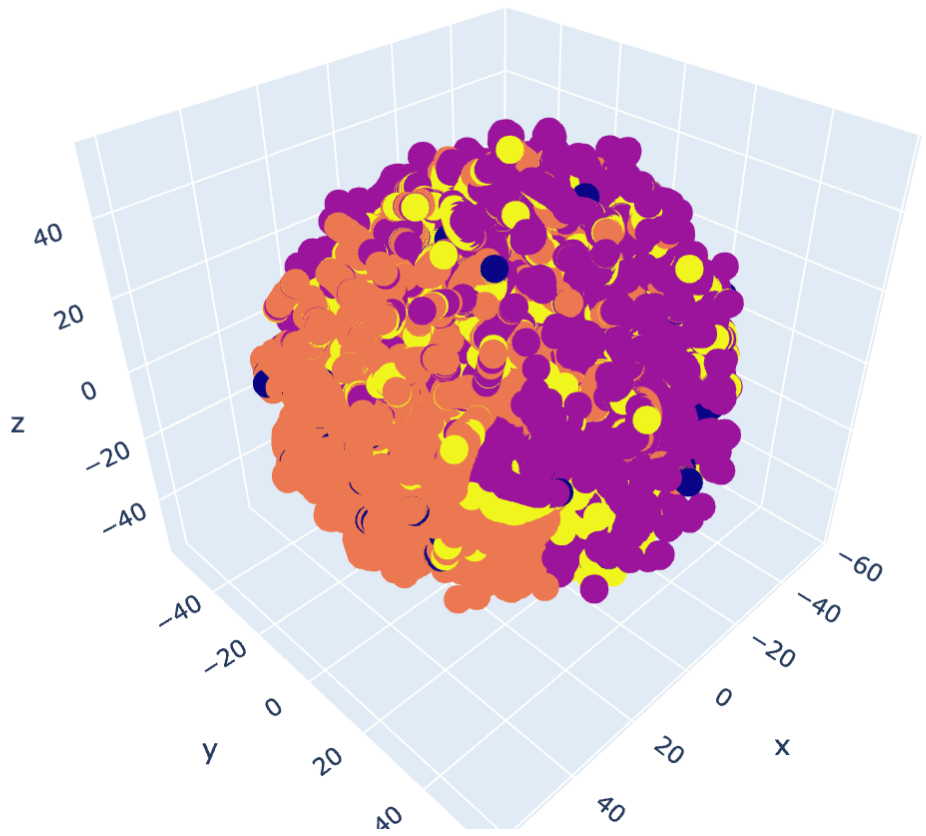


In [126…
```
#TSNE visualization for GMM.

px.scatter(x=X_embedded_tsne[:,0], y=X_embedded_tsne[:,1], width=1000, height=600, color=labels)
```



TSNE gives some sense of clusters. Same densities are being grouped together.

In [128…
```
#3-D visualization(TSNE) for GMM.

px.scatter_3d(x=X_embedded_tsne_3d[:,0], y=X_embedded_tsne_3d[:,1], z=X_embedded_tsne_3d[:,2], width=1000,
```



3-D representation of GMM.

Heirarchical Clustering

In [131…
```
import scipy.cluster.hierarchy as sch
```

In [132…
```
X_scaled_df = pd.DataFrame(X_scaled)
```

In [133…
```
#Performed on a random sample of 5000

X_scaled_df_sampled = X_scaled_df.sample(n=5000)
```

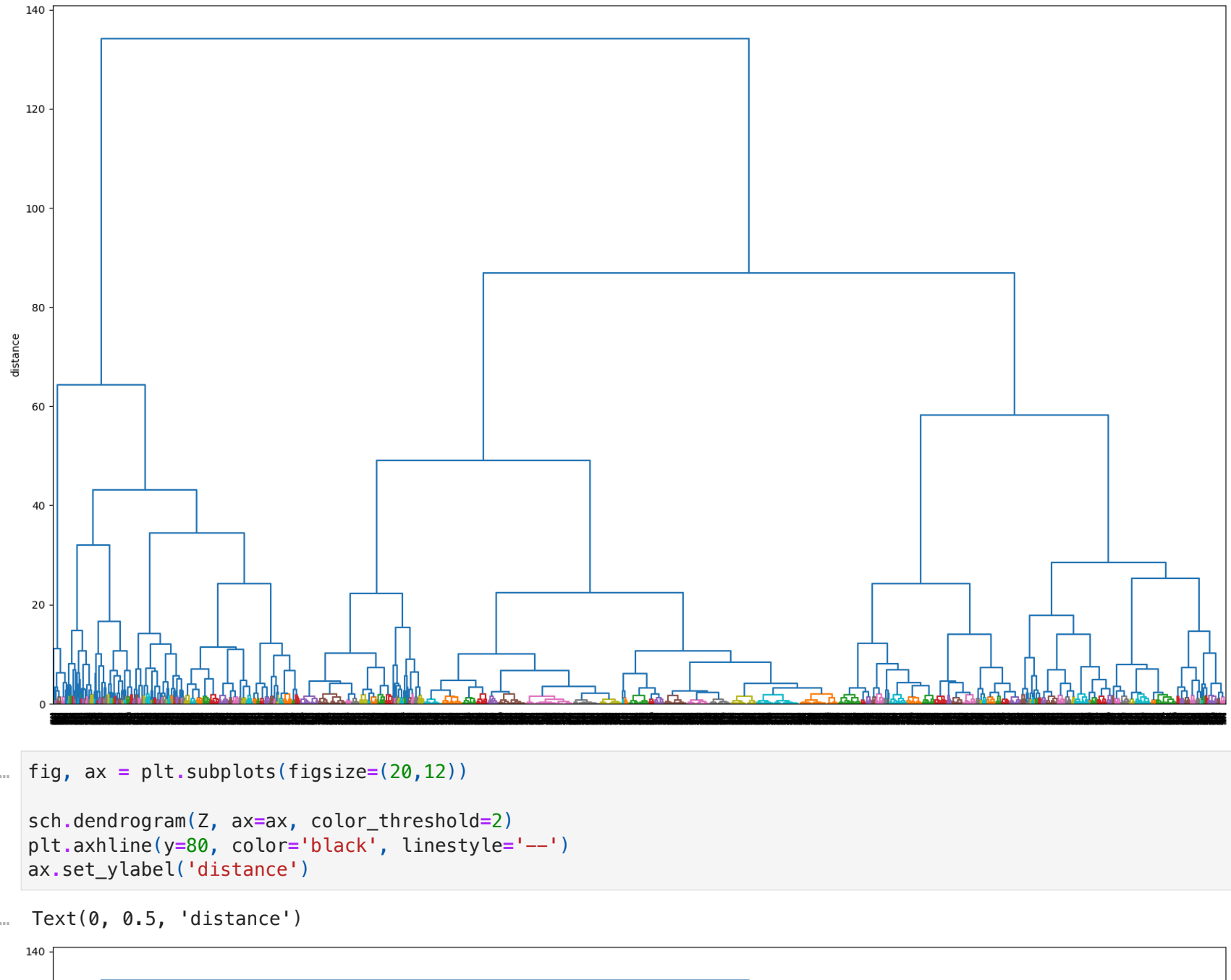
In [134…
```
Z = sch.linkage(X_scaled_df_sampled, method='ward')
```

In [135…
```
Z.shape
```

Out[135… (4999, 4)

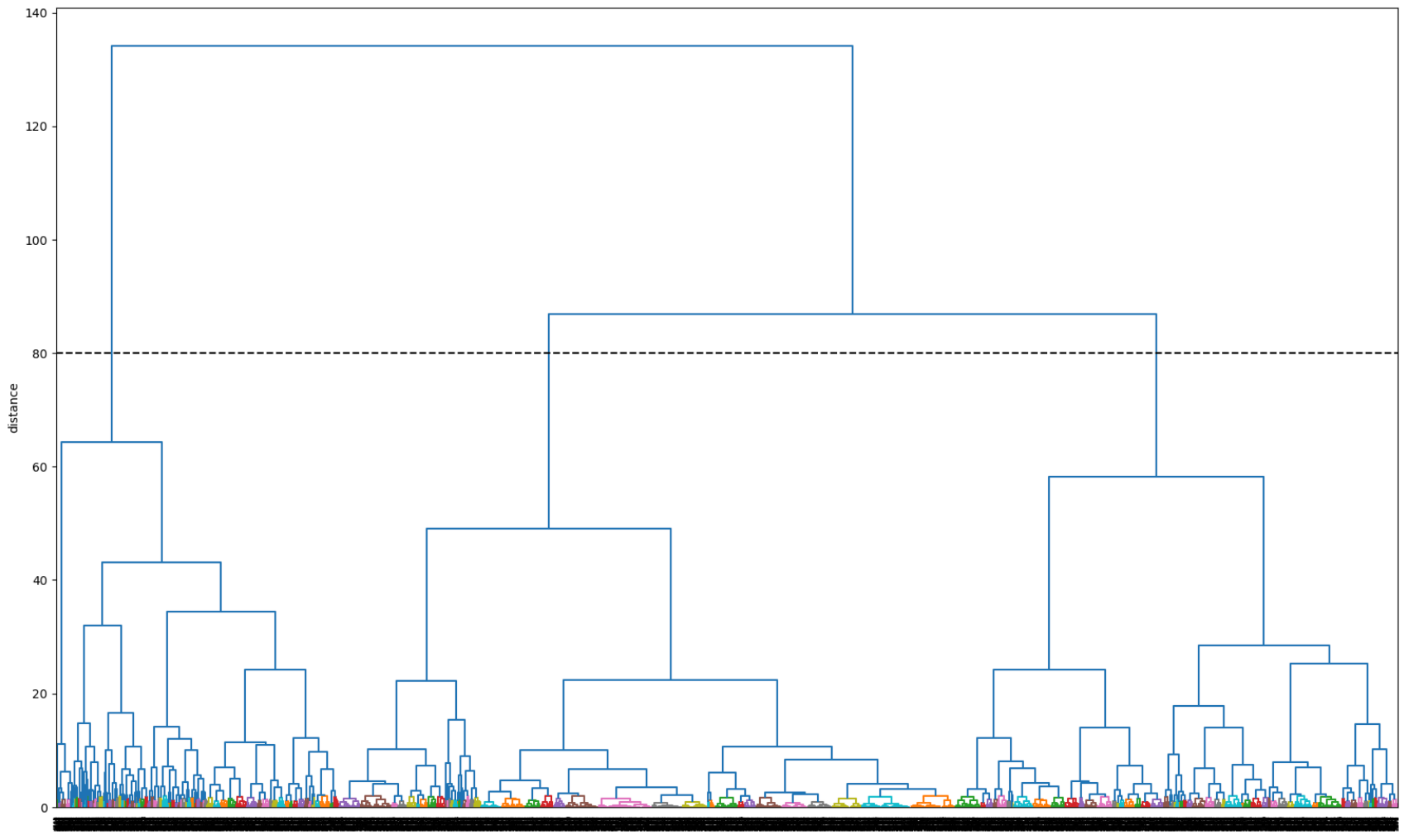
In [136…
```
#Plotting a dendogram for the 5000 samples.

fig, ax = plt.subplots(figsize=(20,12))

sch.dendrogram(Z, ax=ax, color_threshold=2)
ax.set_ylabel('distance')
```

Out[136… Text(0, 0.5, 'distance')



In [137…
```
fig, ax = plt.subplots(figsize=(20,12))

sch.dendrogram(Z, ax=ax, color_threshold=2)
plt.axhline(y=80, color='black', linestyle='--')
ax.set_ylabel('distance')
```

Out[137…
```
Text(0, 0.5, 'distance')
```



We can a good distance between the clusters at k=3.

Clustering with DBSCAN

Given that the data has varying densities(from TSNE), DBSCAN will not be performing very well. Just to observe its behaviour we perform it on a randon sample of 50000.

In [141…
```
from sklearn.cluster import DBSCAN
from sklearn.neighbors import NearestNeighbors
```

In [142…
```
dbscan = X_scaled_df.sample(50000)
```

Hyperparameter tuning - To find the best value of epsilon, we do a k-distance (with k=4) plot and observe the point where we get a sharp increase in distance.

In [144…
```
k = 4
neighbors = NearestNeighbors(n_neighbors=k)
neighbors.fit(dbscan)
```

```
distances, indices = neighbors.kneighbors(dbscan)

sorted_distances = np.sort(distances[:, k-1])
```

In [145…
```
fig = go.Figure()

fig.add_trace(go.Scatter(
    x=np.arange(len(sorted_distances)),  # X-axis will be the sorted indices
    y=sorted_distances,  # Y-axis will be the sorted distances
    mode='lines',
    name=f'{k}-th Nearest Neighbor Distance'
))

fig.update_layout(
    title='k-distance Plot',
    xaxis_title='Points sorted by distance',
    yaxis_title=f'{k}-th Nearest Neighbor Distance',
    template='plotly_dark',
    showlegend=False
)

fig.show()
```

We notice a sharp increase at distance 0.3

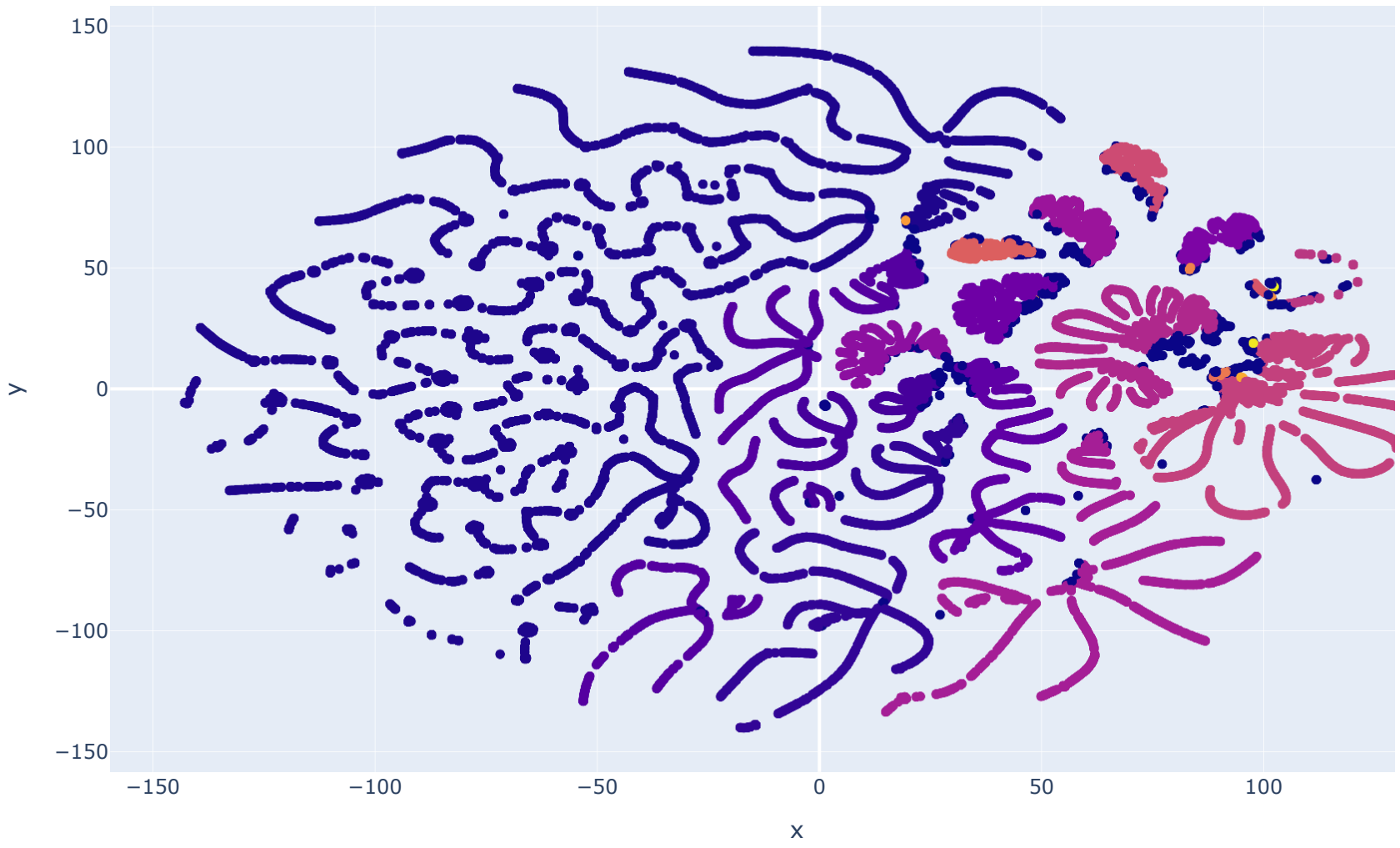
In [147…
```
dbsc = DBSCAN(eps = 0.3, min_samples = 10).fit(dbscan)
```

In [148…
```
dbsclbel = dbsc.labels_
print(dbsclbel)
print(f"Number of clusters: {len(set(dbsclbel))}")
```

```
[0 1 2 ... 0 0 3]
Number of clusters: 28
```

In [149…
```
tsne_2d = TSNE(n_components = 2, perplexity = 50)
dbsc_embed = tsne_2d.fit_transform(dbscan)
```

In [150…
```
px.scatter(x=dbsc_embed[:,0], y=dbsc_embed[:,1], width=1000, height=600, color=dbsclbel)
```



As expected no pattern can be seen in the clusters.

Cluster Interpretation.

In [153…
```
type(km_cluster_final.labels_)
```

Out[153… numpy.ndarray

In [154…
```
X_scaled_df['KMeans_labels'] = km_cluster_final.labels_
```

In [155…
```
X_scaled_df.columns = ['ctc', 'years_of_experience', 'designation', 'class', 'tier', 'KMeans_labels']
X_scaled_df
```

Out[155…

	ctc	years_of_experience	designation	class	tier	KMeans_labels
0	-0.188006	-0.208542	-0.281012	-1.508827	0.515287	0
1	-0.578162	-0.680839	0.932041	0.712087	0.515287	1
2	0.352208	0.027607	-1.494065	-1.508827	0.515287	0
3	-0.428101	-0.444691	0.932041	0.712087	0.515287	1
4	-0.007935	-0.444691	-0.281012	-1.508827	-1.941864	2
…	…	…	…	…	…	…
205618	-0.716216	1.680646	-0.281012	0.712087	0.515287	3
205619	-0.548149	-0.444691	-0.281012	0.712087	0.515287	3
205620	-0.428101	-1.389285	-1.494065	-1.508827	0.515287	0
205621	2.212947	-0.916988	0.932041	0.712087	0.515287	1
205622	-0.103973	0.263755	-1.494065	-1.508827	0.515287	0

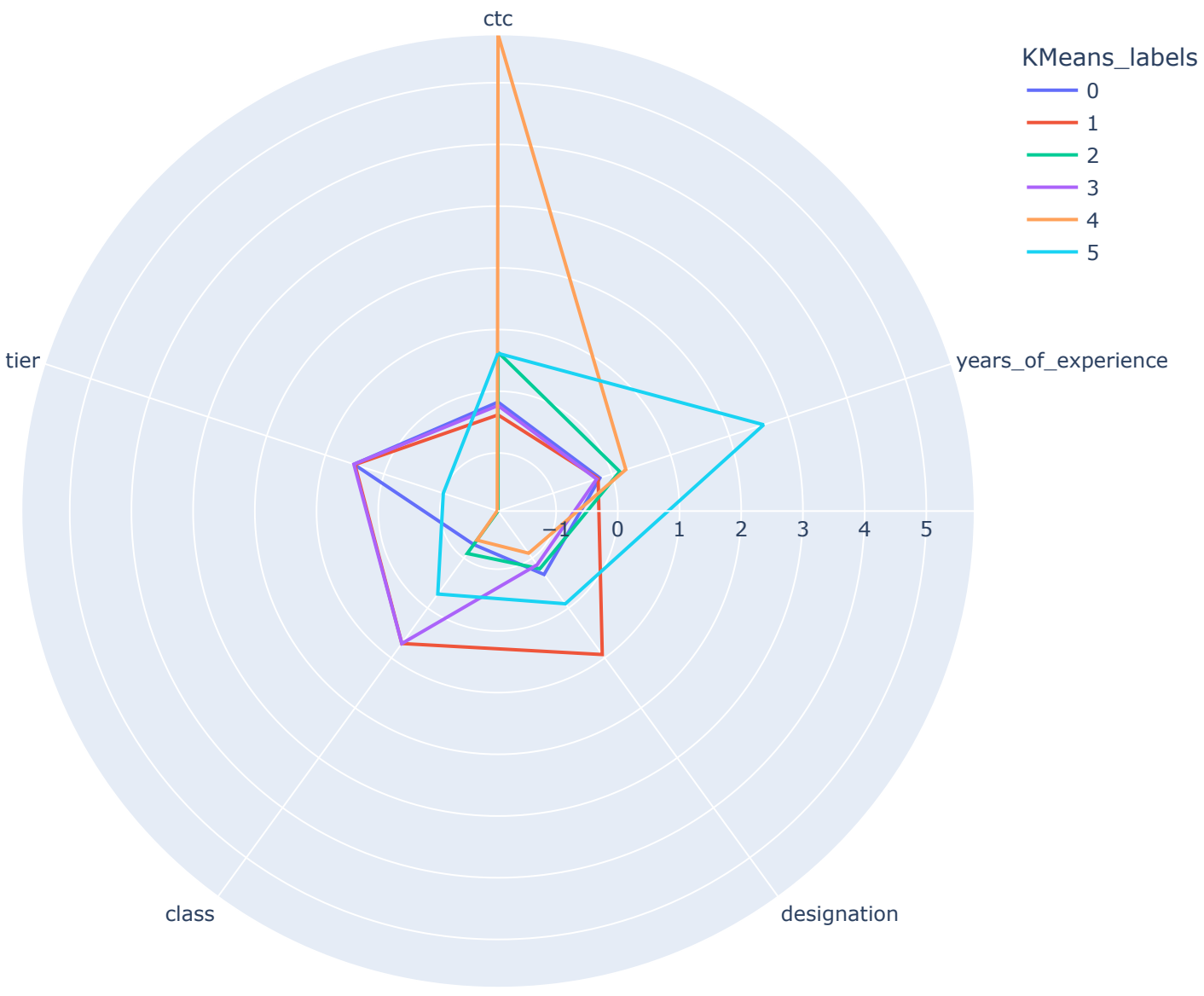
205623 rows × 6 columns

In [156…
```
polar = X_scaled_df.groupby("KMeans_labels").mean().reset_index()
polar = pd.melt(polar, id_vars=["KMeans_labels"])
polar.head(4)
```

Out[156…

	KMeans_labels	variable	value
0	0	ctc	-0.175585
1	1	ctc	-0.382572
2	2	ctc	0.632982
3	3	ctc	-0.231252

In [157…
```
fig = px.line_polar(polar, r="value", theta="variable", color="KMeans_labels", line_close=True,height=700,w
fig.show()
```



Label 0 - Cluster of learners who have low tier, class, designation, ctc and experience. Most probably representing a cluster who are early in their careers.

Label 1 - Cluster of learners who have higher designation, class, and experience but similar tier and ctc as compared to above cluster. Probably in roles where growth is slow.

Label 2 - Cannot interpret.

Label 3 - Cluster of learners who have higher designation, class, and tier but lower experience and ctc. Probably belonging to an industry where ctc growth is low but promotions are frequent.

Label 4 - Looks anomolous, non interpretable.

Label 5 - Cluster of learners who have very high experience and ctc as compared to all others, but have lower designation, class, and tier.