Information about 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 [75]:
```
import re
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
import plotly.figure_factory as ff
import warnings
warnings.filterwarnings("ignore")
from collections import Counter
```

In [77]:
```
import os
os.getcwd()
```

Out[77]: '/Users/phoenix'

In [79]:
```
os.chdir('/Users/phoenix/Downloads')
```

In [81]:
```
df = pd.read_csv('scaler_clustering.csv')
df.head()
```

Out[81]:

	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 [82]:
```
df.shape
```

Out[82]: (205843, 7)

In [83]:
```
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 [86]:
```
df.dtypes
```

Out[86]:
```
Unnamed: 0            int64
company_hash         object
email_hash           object
orgyear             float64
ctc                   int64
job_position         object
ctc_updated_year    float64
dtype: object
```

In [89]:
```
## Checking for null values

df.isnull().sum()
```

Out[89]:
```
Unnamed: 0             0
company_hash          44
email_hash             0
orgyear               86
ctc                    0
job_position       52564
ctc_updated_year       0
dtype: int64
```

In [91]:
```
df["email_hash"].nunique(),df["company_hash"].nunique()
```

Out[91]:
```
(153443, 37299)
```

In [93]:
```
df["email_hash"].value_counts()[:11]
```

Out[93]:
```
email_hash
bbace3cc586400bbc65765bc6a16b77d8913836cfc98b77c05488f02f5714a4b    10
6842660273f70e9aa239026ba33bfe82275d6ab0d20124021b952b5bc3d07e6c     9
298528ce3160cc761e4dc37a07337ee2e0589df251d73645aae209b010210eee     9
3e5e49daa5527a6d5a33599b238bf9bf31e85b9efa9a94f1c88c5e15a6f31378     9
b4d5afa09bec8689017d8b29701b80d664ca37b83cb883376b2e95191320da66     8
faf40195f8c58d5c7edc758cc725a762d51920da996410b80ac4a4d85c803da0     8
4818edfd67ed8563dde5d083306485d91d19f4f1c95d193a1700e79dd245b75c     8
c0eb129061675da412b0deb15871dd06ef0d7cd86eb5f7e8cc6a20b0d1938183     8
d598d6f1fb21b45593c2afc1c2f76ae9f4cb7167156cdf93246d4192a89d8065     8
d15041f58bb01c8ee29f72e33b136e26bc32f3169a40b53d75fe7ae9cbb9a551     8
aacf9473e3cee3e3f7c322e49bb8a6d5346bb05f3ff5bb9e9ac3ae22729ab933     7
Name: count, dtype: int64
```

In [95]:
```
## Removing Special Characters Using Regex

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[95]:
```
'airtel x labs'
```

In [97]:
```
df['job_position'] = df['job_position'].apply(lambda x: preprocess_string(str(x)))
df['job_position'].nunique()
```

Out[97]:
```
856
```

In [98]:
```
df['company_hash'] = df['company_hash'].apply(lambda x: preprocess_string(str(x)))
df['company_hash'].nunique()
```

Out[98]:
```
37208
```

In [100…
```
df.shape
```

Out[100… (205843, 7)

In [103…
```
## Dropping Unnecessary Columns

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

In [105…
```
## Dropping Duplicates

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

Out[105… (205697, 6)

In [106…
```
## Concerting 'nan' to 'NaN' value

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

In [109…
```
df
```

Out[109…

	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

In [111…
```
## Imputing NaN Values by grouping on Company hash and CTC

def temp(lst):
    """
    Finds the most frequently occurring element in a list.

    Parameters:
    lst (list): A list of elements.

    Returns:
    any: The element that appears most frequently in the list.
         If multiple elements have the same highest frequency, the first encountered is returned.
    """
    d = Counter(lst)
    return max(d, key=d.get)

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

In [112…
```
df.isnull().sum()
```

Out[112…
```
company_hash           44
email_hash              0
orgyear                86
ctc                     0
job_position        37126
ctc_updated_year        0
dtype: int64
```

In [113…
```
df
```

Out[113…

	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	NaN	
205841	zgn vuurxwvmrt	fb46a1a2752f5f652ce634f6178d0578ef6995ee59f6c8...	2019.0	5100000	NaN	
205842	bgqsvz onvzrtj	0bcfc1d05f2e8dc4147743a1313aa70a119b41b30d4a1f...	2014.0	1240000	NaN	

205697 rows × 6 columns

In [114…

```
## Replacing missing values with Others.

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

In [115…

```
## Grouping on company hash and job position and finally finding mean CTC for each job position

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

Out[115…

	company_hash	job_position	orgyear	ctc	ctc_updated_year
0	Others	android engineer	2014.0	4.800000e+05	2020.000000
1	Others	backend engineer	2015.5	8.165714e+05	2019.571429
2	Others	data analyst	2019.5	2.100000e+06	2018.500000
3	Others	data scientist	2020.0	2.100000e+06	2019.000000
4	Others	engineering intern	2019.5	7.750000e+05	2019.500000
...	...	...	...	...	...
58451	zyvzwt wgzohrnxzs tzsxzttqo	frontend engineer	2012.0	9.400000e+05	2019.000000
58452	zz	other	2013.0	1.370000e+06	2020.000000
58453	zz wgzztwn mya	other	2009.0	2.380000e+04	2017.000000
58454	zzb ztdnstz vacxogqj ucn rna	fullstack engineer	2017.0	6.000000e+05	2021.000000
58455	zzzbzb	other	1990.0	7.200000e+05	2020.000000

58456 rows × 5 columns

In [121…

```
## Define function to impute missing job positions

def impute_job_position(row, avg_ctc_by_position):
    """
    Imputes the missing job position based on the closest CTC within the same company.

    Parameters:
    row (pd.Series): A row of the DataFrame containing "job_position", "company_hash", and "ctc".
    avg_ctc_by_position (pd.DataFrame): A DataFrame containing average CTC values by job position and compa

    Returns:
        - The imputed job position if "job_position" is missing.
        - The original job position if it is not missing.
    """
    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"]

## Apply the function to fill NaN values
df["job_position"] = df.apply(lambda row: impute_job_position(row, avg_ctc_by_pos), axis=1)

print(df)
```

```
                  company_hash  \
0                 atrgxnnt xzaxv
1       qtrxvzwt xzegwgbb rxbxnta
2                 ojzwnvwnxw vx
3                     ngpgutaxv
4                    qxen sqghu
...                         ...
205838                vuurt xzw
205839                husqvawgb
205840                 vwwgrxnt
205841            zgn vuurxwvmrt
205842            bgqsvz onvzrtj

                                             email_hash  orgyear      ctc  \
0       6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...   2016.0  1100000
1       b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...   2018.0   449999
2       4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...   2015.0  2000000
3       effdede7a2e7c2af664c8a31d9346385016128d66bbc58...   2017.0   700000
4       6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...   2017.0  1400000
...                                                   ...      ...      ...
205838  70027b728c8ee901fe979533ed94ffda97be08fc23f33b...   2008.0   220000
205839  7f7292ffad724ebbe9ca860f515245368d714c84705b42...   2017.0   500000
205840  cb25cc7304e9a24facda7f5567c7922ffc48e3d5d6018c...   2021.0   700000
205841  fb46a1a2752f5f652ce634f6178d0578ef6995ee59f6c8...   2019.0  5100000
205842  0bcfc1d05f2e8dc4147743a1313aa70a119b41b30d4a1f...   2014.0  1240000

                      job_position  ctc_updated_year
0                            other            2020.0
1                fullstack engineer            2019.0
2                  backend engineer            2020.0
3                  backend engineer            2019.0
4                fullstack engineer            2019.0
...                             ...               ...
205838                          NaN            2019.0
205839           fullstack engineer            2020.0
205840                 qa engineer             2021.0
205841            frontend engineer            2019.0
205842  associate software developer            2016.0

[205697 rows x 6 columns]
```

In [123…
```
df.isna().sum()
```

Out[123…
```
company_hash          44
email_hash             0
orgyear               86
ctc                    0
job_position        4045
ctc_updated_year       0
dtype: int64
```

In [125…
```
df
```

Out[125…

	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 [127…
```
#Filling null values with others

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

In [129…
```
df.isna().sum()
```

Out[129…
```
company_hash         0
email_hash           0
orgyear             86
ctc                  0
job_position         0
ctc_updated_year     0
dtype: int64
```

In [131…
```
df["orgyear"].value_counts()
```

Out[131…
```
orgyear
2018.0    25248
2019.0    23416
2017.0    23214
2016.0    23012
2015.0    20589
          ...
2107.0        1
1972.0        1
2101.0        1
208.0         1
200.0         1
Name: count, Length: 77, dtype: int64
```

In [133…
```
## Assigning Company name with count less than 1 with 'Others'

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

In [135…
```
df['company_hash'].nunique()
```

Out[135…
```
3779
```

In [137…
```
df["orgyear"].value_counts()
```

Out[137…
```
orgyear
2018.0    25248
2019.0    23416
2017.0    23214
2016.0    23012
2015.0    20589
          ...
2107.0        1
1972.0        1
2101.0        1
208.0         1
200.0         1
Name: count, Length: 77, dtype: int64
```

In [139…
```
dictio = dict(df.groupby('company_hash')["orgyear"].median())
dictio
```

Out[139… {'Others': 2015.0,
'a ntwyzgrgsxto': 2015.0,
'aaqxctz avnv owxtzwto vzvrjnxwo ucn rna': 2016.0,
'abwavnv ojontb': 2011.0,
'adw ntwyzgrgsj': 2015.0,
'adw ntwyzgrgsxto': 2015.0,
'agdutq': 2016.5,
'aggartmrht xzzgcvnxgzo': 2014.0,
'aghmnzhn': 2017.0,
'agnoihvqto': 2013.5,
'agnut': 2018.0,
'agnutxz': 2019.0,
'agotrtwn': 2017.0,
'agrwtqv': 2015.0,
'agwgz ntwyzgrgsxto': 2017.5,
'agwihxnj': 2017.0,
'agwovuu wgzohrn v outwxvrxon agwngq xz bxzhnto': 2016.0,
'agzn fgqp xz vzj gqsvzxkvnxgz': 2019.0,
'ahwpwqttp': 2017.0,
'ahzkg': 2016.0,
'ahzzyhbmj': 2015.0,
'ahzzyhbmj xn otqcxwto': 2012.0,
'ajzvbxw axsxnvr ogrhnxgzo unt rna': 2016.5,
'ajzvnv': 2016.0,
'amgx': 2010.0,
'amo mvzp': 2016.0,
'amo voxv yhm': 2015.5,
'amo voxv yhm ucn rna': 2016.0,
'anww': 2001.0,
'aqag mvzsvrgqt': 2017.0,
'aqggb ntwyzgrgsj': 2017.0,
'aqgumgd': 2011.0,
'aqhcv': 2015.0,
'aqhcv avnv': 2010.0,
'aqtvb': 2015.0,
'aqtvbgqmxn': 2014.5,
'aqvhu': 2016.0,
'aqxctkj': 2017.0,
'aqxoynx ogen': 2015.5,
'aqxoynx ogen ogrhnxgzo': 2013.5,
'aqxoynxogen ucn rna': 2016.5,
'arn rvmo': 2019.0,
'asqvuy rvmo': 2017.0,
'atbvzaevqb': 2014.0,
'atcntonwg ihgnt': 2014.0,
'atcorvzt': 2019.0,
'atctrgubtzn mvzp ge oxzsvugqt': 2015.0,
'atetzwt qtotvqwy vza atctrgubtzn gqsvzxovnxgz': 2019.0,
'atf ogrhnxgzo ucn rna': 2016.0,
'athnowyt mvzp': 2018.0,
'athnowyt ntrtpgb': 2016.0,
'athonwyt mvzp': 2020.0,
'atigat': 2018.0,
'atrgxnnt': 2016.0,
'atrgxnnt axsxnvr': 2014.0,
'atrgxnnt ho xzaxv': 2017.0,
'atrgxnnt nghwyt': 2020.0,
'atrgxnnt wgzohrnxzs xzaxv uqxcvnt rxbxnta': 2015.0,
'atrgxnnt xzaxv': 2014.0,
'atrgxnnt xzaxv geexwto ge nyt ho': 2017.0,
'atrhdt': 2016.0,
'atrnv trtwnqgzxwo': 2018.0,
'atrnv vxq rxzto': 2015.0,
'atrr': 2016.0,
'atrr ntwyzgrgsxto': 2016.0,
'atrr tbw': 2014.0,
'atrvurtd': 2015.0,
'atrxctqj ytqg': 2013.0,
'atryxctqj': 2018.0,
'atturgsxwntwy': 2019.0,
'attutz vx': 2017.0,
'attuxzntzn': 2018.0,
'atvroyvqt': 2016.0,
'atvrtqogwptn': 2017.0,
'atwhqnxo wgqugqvnxgz': 2015.0,
'atwvnyrgz ougqno xzaxv': 2016.0,
'atwxbvr ntwyzgrgsxto': 2017.0,
'atwxoxgz bxzao': 2013.0,
'atwxoxgz qtoghqwto sqghu': 2014.5,
'atznoh vtsxo ztnfgqp': 2017.0,
'au fgqra': 2017.5,
'aurb ogenfvqt ogrhnxgzo rxbxnta': 2015.0,

```
'avbwg ogrhnxgzo': 2014.5,
'aveegaxr ogenfvqt': 2017.0,
'avnv otbvznxwo': 2015.0,
'avnvbtnxwv ogrhnxgzo': 2018.0,
'avnvbtnxwv ogrhnxgzo uqxcvnt rxbxnta': 2015.0,
'avnvbvnxwo srgmvr otqcxwto': 2012.0,
'avnvbvnxwo srgmvr otqcxwto rxbxnta': 2014.5,
'avnvbx': 2017.0,
'avnvcvr vzvrjnxwo wgqugqvnxgz': 2017.0,
'avnvdh': 2017.0,
'avnveghzaqj': 2011.0,
'avnvftvct ucn rna': 2016.5,
'avnvrxwxgho': 2015.0,
'avnvshxot': 2014.0,
'avnvsqgpq': 2017.5,
'avnvwgqt ogenfvqt': 2020.0,
'avoovhrn ojonbto': 2016.0,
'avoovhrn ojontbto xz': 2016.0,
'avowti': 2019.0,
'avqfxzmgd': 2015.0,
'avqp ygqot': 2018.0,
'avqpygqot axsxnvr ogrhnxgzo': 2018.0,
'avxrjyhzn': 2017.0,
'avxrjzxzlv': 2015.0,
'avxxwn': 2019.0,
'avxzxp myvopvq': 2012.0,
'avzopt xn': 2014.5,
'awm mvzp': 2016.0,
'axbno': 2013.0,
'axbtzoxgz avnv': 2015.5,
'axgz srgmvr ogrhnxgzo rna': 2018.0,
'axowgctq agrrvq': 2015.0,
'axoy ztnfgqp': 2014.0,
'axpoyv ntwyzgrgsxto': 2013.0,
'axqtwnx': 2016.0,
'axsxn xzohqvzwt': 2015.0,
'axsxnt xzegntwy': 2020.0,
'axsxnvr bwpxzotj': 2018.0,
'axsxnvr shvqaxvz': 2013.0,
'axsxnvr sqttz': 2017.5,
'axsxnvr yvqmgq': 2004.0,
'axtmgra zxdagqe': 2013.0,
'axvouvqp xzw': 2017.0,
'axvruva': 2015.5,
'axztghn': 2012.5,
'ayr': 2013.5,
'azvnv': 2015.0,
'b y vroyvjv wg': 2016.0,
'bd urvjtq': 2014.0,
'bexzt': 2016.0,
'bgatr z': 2014.0,
'bgavp vzvrjnxwo': 2018.0,
'bgbouqtoog': 2014.0,
'bgctxzojzw': 2016.0,
'bggajo vzvrjnxwo': 2013.0,
'bggrjv ntonxzs': 2014.0,
'bggzeqgs rvmo': 2015.5,
'bggzo ntwy rna': 2016.0,
'bggzqven xzzcgcvnxgzo rvm': 2015.0,
'bggzqven xzzgcvnxgz rvmo': 2015.0,
'bghzn mrht ntwyzgrgsxto': 2017.5,
'bghznmrht ntwyzgrgsxto': 2018.5,
'bglg ztnfgqpo xzw': 2015.0,
'bgmonvw': 2019.0,
'bgmxgnxwo xn ogrhnxgz ucn rna': 2016.0,
'bgmxihxnj xzw': 2011.0,
'bgmxogen xzegntwy': 2015.5,
'bgmxoj ntwyzgrgsj': 2017.0,
'bgmxpfxp': 2015.0,
'bgmxpvov': 2013.0,
'bgmxrgxnnt': 2016.0,
'bgmxrgxnnt ntwyzgrgsxto': 2018.0,
'bgmxrt uqgsqvbbxzs rrw': 2015.0,
'bgmxrt uqtbxtq rtvsht': 2018.0,
'bgmxrtfvrrv': 2021.0,
'bgmxrthb': 2014.0,
'bgmxrto': 2013.5,
'bgmxrtwgatqk': 2017.0,
'bgmxrtxqgz': 2011.0,
'bgmxrxjv': 2013.5,
'bgmxrxjv ntwyzgrgsxto': 2013.0,
'bgngqgrv bgmxrxnj v rtzgcg wgbuvzj': 2015.0,
'bgngqgrv ogrhnxgzo': 2015.0,
```

```
'bgngqi': 2017.0,
'bgnxcxnj rvmo': 2016.0,
'bgnytqogzohbx xzegntwy  atoxszo': 2015.0,
'bgnytqogzohbx xzegntwy  atoxszo rna bxza': 2017.0,
'bgqsvz onvzrtj': 2015.0,
'bgqtjtvyo': 2017.0,
'bgqzxzsonvq': 2016.0,
'bgqzxzsonvq xzaxv ucn rna': 2005.0,
'bgsrxd': 2017.0,
'bgtzsvst': 2015.0,
'bgtzsvst xzw': 2015.0,
'bgzgnjut': 2011.0,
'bgzgnjut ogrhnxgzo xzaxv': 2017.0,
'bgzgnjut xbvsxzs': 2017.0,
'bgzgwtun': 2016.0,
'bgzontq': 2015.5,
'bgztj cxtf': 2016.0,
'bgztjngq': 2016.0,
'bgztjnvu': 2015.5,
'bh oxsbv': 2018.0,
'bh oxsbv mhoxztoo ogrhnxgzo': 2015.0,
'bh oxsbv xzw': 2017.0,
'bhcxwgb': 2015.0,
'bhnhvr bgmxrt': 2018.0,
'bhrnxwgqtfvqt': 2018.0,
'bj vrrj': 2017.0,
'bjbgztjpvqbv': 2016.0,
'bjgutqvngq': 2017.0,
'bjobvqnuqxwtwgb': 2014.0,
'bjouxzzj': 2018.0,
'bjowyggruvst': 2019.0,
'bjpvvqbv': 2016.0,
'bjsrvbb': 2015.0,
'bjsvnt': 2016.0,
'bjvacgxz': 2016.0,
'bjvzvngbj': 2019.0,
'bjvzvngbj xzntsqvnxgz ucn rna': 2020.0,
'bjwgb gox': 2010.0,
'bjwgbgox': 2011.0,
'bjznqv': 2014.0,
'bjznqv lvmgzs': 2016.0,
'bjznqvlvmgzs': 2016.0,
'bjznqvwgb': 2012.0,
'bmqax': 2017.0,
'bnd sqghu xzw': 2018.0,
'bnqtt ogenfvqt ucn rna': 2019.0,
'bojo ntwyzgrgsxto rrw': 2015.0,
'bos srgmvr ogrhnxgzo': 2015.0,
'bosvx': 2014.0,
'bowqxuno': 2019.0,
'bowx xzw': 2014.0,
'bq wggutq': 2017.0,
'bqqogen': 2014.0,
'bqwggutq': 2016.5,
'bs': 2017.0,
'bs ntwyzgrgsxto ucn rna': 2017.0,
'btactqct ytvrnywvqt uqxcvnt rxbxnta': 2016.0,
'btanqgzxw': 2013.0,
'btarxet xzntqzvnxgzvr uqxcvnt rxbxnta': 2016.0,
'btarxetwgb': 2015.0,
'btaurho btaxwxzv uqtuvsvav ov': 2018.0,
'btawgqao': 2016.5,
'btax vooxon xzaxv ucn rna': 2016.0,
'btaxmhaaj': 2018.5,
'btaxnvm ogenfvqt xzaxv ucn rna': 2018.0,
'btaxnvm ogenfvqt xzw': 2018.5,
'btaxv btrgz': 2018.0,
'btaxv xi axsxnvr rna': 2013.0,
'btaxvgwtvz': 2016.0,
'btaxvgwtvz voxv ucn rna': 2011.0,
'btaxvntp': 2015.0,
'btaxvntp xzaxv ntwyzgrgsj ucn rna': 2014.0,
'btaxvntp xzaxv uqxcvnt rxbxnta': 2018.5,
'btaxvpxza': 2016.0,
'btaxvvsxrxnj': 2014.0,
'btaxvztn': 2017.0,
'btaxvztnaxqtwnx': 2018.0,
'btnnr': 2014.5,
'btnnrt ntwy': 2015.0,
'btnqg wvoywvqqj': 2019.0,
'btnqxwonqtvb': 2018.0,
'btnrxet': 2015.0,
'btnvatoxsz ogrhnxgzo': 2014.5,
```

```
'btnvwhmt ogenfvqt': 2019.0,
'btnvwhmt ogenfvqt ucn rna': 2016.0,
'btqh wvm wgbuvzj ucn rna': 2009.0,
'btqprt ogpqvnx': 2020.0,
'btqtaxny': 2013.0,
'btqtaxny wgqugqvnxgz': 2013.5,
'btqwtato mtzk': 2013.0,
'btqwtato mtzk qtotvqwy vza atctrgubtzn xzaxv': 2015.0,
'btqwtatomtzk qtotvqwy vza atctrgubtzn xzaxv': 2016.0,
'btqwtq': 2015.0,
'btqwtq wgzohrnxzs xzaxv ucn rna': 2014.0,
'btqwtq xzaxv': 2013.5,
'btqxrjnxwo': 2018.0,
'btrcvhrn': 2015.0,
'btrgqqv': 2015.0,
'bttoyg': 2015.0,
'btvohqtgzt': 2017.0,
'btwy bgwyv': 2018.0,
'btzngq sqvuyxwo': 2016.0,
'btzngq sqvuyxwo ucn  rna': 2014.0,
'btznxgz utgurt xz jghq ugono': 2018.0,
'buo rxbxnta': 2015.0,
'bur': 2018.0,
'buyqd': 2016.0,
'buyvoxo': 2016.0,
'buyvoxo rna': 2016.0,
'buyvoxo ucn rna': 2016.0,
'bvctqxw ojontbo': 2016.0,
'bvctz fvct': 2017.0,
'bvctzxq': 2016.0,
'bvd rxet xzohqvzwt': 2016.0,
'bvdxb xzntsqvnta': 2015.5,
'bvdxbr': 2018.0,
'bvdxbr rvmo': 2019.0,
'bvi ogenfvqt': 2018.5,
'bvjmvzp': 2017.0,
'bvjvavnv': 2019.0,
'bvltowg': 2015.0,
'bvnwybgct uvj ucn rna': 2015.0,
'bvnyfgqpo': 2018.0,
'bvnyfgqpo xzaxv ucn rna': 2018.0,
'bvon srgmvr': 2020.0,
'bvontp rna': 2016.0,
'bvontqwvqa': 2014.0,
'bvovx owyggr': 2019.0,
'bvoyqti mvzp': 2010.0,
'bvptbjnqxu': 2016.0,
'bvptbjnqxuwgb': 2014.0,
'bvqctrr otbxwgzahwngq': 2011.5,
'bvqctrr otbxwgzahwngqo': 2014.0,
'bvqgugon': 2017.0,
'bvqhnx ohkhpx xzaxv rxbxnta': 2016.0,
'bvqihxo ntwyzgrgsxto': 2016.0,
'bvqoy  bwrtzzvz wgbuvzxto': 2018.0,
'bvqptn oxburxexta': 2011.0,
'bvqpxn': 2009.0,
'bvqpxn xzaxv otqcxwto ucn rna': 2011.0,
'bvqrvmo ogenfvqt': 2013.5,
'bvqrvmo xzw': 2014.5,
'bvsxw ogenfvqt xzw': 2015.0,
'bvsxwmqxwpowgb': 2012.5,
'bvsxwuxz ovbvon ntwyzgrgsxto': 2018.0,
'bvsztnx bvqtrrx': 2016.0,
'bvszv xzegntwy  v axcxoxgz ge ihtoo wgqu rxbxnta': 2013.0,
'bvszvihton ntwyzgrgsxto': 2015.0,
'bvszxnhat ogenfvqt xzw': 2018.0,
'bvtqop': 2011.0,
'bvtqop srgmvr otqcxwt wtznqto': 2011.0,
'bvubjxzaxv': 2019.0,
'bvuq ntwyzgrgsxto': 2014.0,
'bvurxzt': 2018.0,
'bvwihvqxt sqghu': 2011.5,
'bvwjo': 2010.0,
'bvyvqvoynqv pzgfrtast wgqugqvnxgz rxbxnta bpwr': 2019.0,
'bvyxzaqv  bvyxzaqv': 2017.0,
'bvyxzaqv  bvyxzaqv rna': 2015.0,
'bvyxzaqv qxot': 2015.5,
'bvyxzaqv vza bvyxzaqv': 2017.5,
'bvyxzaqv wgbcxcv': 2018.0,
'bvyxzaqv wgbcxcv ntwyzgrgsxto rna': 2016.5,
'bvznqv rvmo': 2016.0,
'bvznyvz': 2015.0,
'bvzsgvuuo': 2016.0,
```

```
'bvzugftqsqghu': 2014.0,
'bvzxn myguvr': 2020.0,
'bvzyvnnvz voogwxvnto': 2016.0,
'bvzz xzaxv': 2017.0,
'bvzzvzvx': 2016.0,
'bwevajtz axsxnvr': 2016.0,
'bwpxzotj': 2015.5,
'bwpxzotj  wgbuvzj': 2017.0,
'bwpxzotj vza wgbuvzj': 2014.0,
'bwv onhatzn': 2019.0,
'bwvett': 2012.0,
'bwvett ogenfvqt xzaxv uqxcvnt rxbxnta': 2011.0,
'bxi': 2017.0,
'bxi axsxnvr': 2017.0,
'bxnqvntwy': 2016.5,
'bxntr': 2015.0,
'bxojo': 2012.0,
'bxojo ogenfvqt ogrhnxgzo': 2015.5,
'bxonqvr ogrhnxgzo ucn rna': 2017.0,
'bxqvwrt ogenfvqt ojontbo xzw': 2017.0,
'bxqvznxo': 2006.5,
'bxrcxp ntwyzgrgsj otqcxwto xzaxv ucn rna': 2017.0,
'bxrpmvoptn': 2015.0,
'bxrrxbvz': 2015.0,
'bxrvvu': 2016.0,
'bxwqg egwho': 2016.0,
'bxwqg egwho ogenfvqt atctrgubtzn': 2018.0,
'bxwqgbvd xzegqbvnxwo rna': 2012.0,
'bxwqgogen': 2015.0,
'bxwqgogen qtotvqwy xzaxv': 2017.0,
'bxwqgogen wgqugqvnxgz': 2012.0,
'bxwqgogen xaw': 2017.0,
'bxwqgonqvntsj': 2018.0,
'bxwqgotbx wgqugqvnxgz': 2012.0,
'bxwqgrvza rxbxnta': 2016.0,
'bxwqgwyxu ntwyzgrgsj': 2012.5,
'bxwqgwyxu ntwyzgrgsj xzw': 2016.0,
'bxwqgz ntwyzgrgsj': 2016.0,
'bxyhu wgbbhzxwvnxgz': 2019.0,
'bxzaexqt ogrhnxgzo': 2017.0,
'bxzanqtt': 2016.0,
'bxzantwp': 2016.0,
'bxzanxwprt': 2016.0,
'bxzaonxd ogenfvqt rvmo': 2016.0,
'bxzasvnt ogrhnxgz ucn rna': 2014.0,
'bxzxgqvzst': 2019.0,
'bzw rna': 2016.0,
'bzw wgbuvzj': 2012.0,
'bzzxn vrrvyvmva': 2019.0,
'cbfvqt': 2014.0,
'cbfvqt vxqfvnwy': 2011.0,
'cbgwp': 2017.0,
'ccaz ntwyzgrgsxto': 2016.0,
'cgavegzt': 2013.0,
'cgavegzt cgxo': 2016.0,
'cgavegzt oyvqta otqcxwto rxbxnta': 2016.0,
'cgavegzt xatv': 2016.0,
'cgavegzt xzaxv': 2016.5,
'cgavegzt xzaxv otqcxwto ucn rna': 2012.5,
'cggzxp': 2015.0,
'cgjrrv evoyxgzo ucn rna': 2018.0,
'cgrcg sqghu': 2013.0,
'cgrpofvstz': 2014.0,
'cgrpofvstz xn otqcxwt xzaxv ucn rna': 2017.0,
'cgrpofvstz xn otqcxwto xzaxv': 2015.0,
'cgrvznt': 2020.0,
'cgrvznt ntwyzgrgsxto': 2017.0,
'cgrvzojo ntwyzgrgsxto': 2016.5,
'cgsg qtznvro': 2015.0,
'cgtqtxq vm': 2018.0,
'cgwtqv wgbbhzxwvnxgzo': 2014.0,
'chwrxu xzw': 2012.5,
'cjbg': 2017.0,
'cjbg ogrhnxgzo ucn rna': 2015.5,
'cjgb rvmo': 2015.0,
'cjvuvq': 2016.0,
'cogen ntwyzgrgsxto ucn rna': 2016.0,
'cogen wgzohrnxzs sqghu xzw': 2015.0,
'cotqc mhoxztoo ogrhnxgz': 2015.0,
'crgwxnj': 2013.0,
'crgwxnj wrgha vuurxwvnxgzo': 2015.0,
'crxzp xzw': 2013.0,
'ctav rvmo': 2018.0,
```

```
'ctavznh': 2017.0,
'ctavznv qtoghqwto urw': 2017.0,
'ctbmh ntwyzgrgsxto': 2015.0,
'ctqnvegqt': 2015.5,
'ctqot': 2012.5,
'ctqov ztnfgqpo': 2013.0,
'ctqv otwhqxnj': 2012.0,
'ctqxegzt': 2015.0,
'ctqxkgz': 2015.0,
'ctqxkgz avnv otqcxwto': 2013.0,
'ctqxkgz btaxv': 2011.0,
'ctqxkgz wgzztwn': 2013.0,
'ctqxkgz xzaxv': 2014.0,
'ctqxnvo ntwyzgrgsxto rrw': 2015.5,
'ctqxop vzvrjnxwo': 2014.5,
'ctqxznojontbo': 2015.0,
'ctrgnxg ntwyzgrgsxto': 2016.0,
'ctrgwxo': 2013.0,
'ctrgwxo ojontb ucn rna': 2010.0,
'ctrvzpvzx ogenfvqt': 2017.0,
'ctrvzpvzx wgbbhzxwvnxgzo ntwyzgrgsxto': 2015.0,
'ctwngowvrvq': 2019.0,
'ctwngowvrvq ntwyzgrgsxto uqxcvnt rxbxnta': 2015.0,
'ctwngq xzegqbvnxp': 2014.0,
'ctytqt': 2018.0,
'ctyvzn ntwyzgrgsxto egqbtqrj pqxnxpvr otwhqtowvz': 2019.0,
'ctztqv ntwyzgrgsxto': 2018.0,
'cvjvzv': 2015.0,
'cvjvzv ztnfgqpo': 2017.0,
'cvoovq rvmo': 2019.0,
'cvqxvz btaxwvr ojontbo': 2014.0,
'cvrhtbgbtznhb': 2018.0,
'cvrhtbgbtznhb ogenfvqt otqcxwto': 2014.0,
'cvrhtej ogrhnxgzo': 2018.0,
'cvrhtexqon axsxnvr btaxv uqxcvnt rxbxnta': 2016.5,
'cvrhtrvmo': 2015.0,
'cvrhwgzztd ntrtwgb qtrxtzwt lxg': 2015.0,
'cvrntwy wgqugqvnxgz': 2017.0,
'cvrtg': 2015.0,
'cvrtg xzaxv': 2014.0,
'cvwg mxzvqj otbvznxwo rru': 2016.0,
'cxaggrj': 2019.0,
'cxatgptz': 2016.0,
'cxbtg': 2017.0,
'cxcg bgmxrt xzaxv ucn rna': 2015.0,
'cxkhqj': 2015.0,
'cxktdutqno': 2015.0,
'cxnqvzv': 2019.0,
'cxntgo ehza otqcxwto': 2013.0,
'cxo wvqttqo': 2014.0,
'cxontd': 2015.0,
'cxontgz ntwyzxwvr vza otqcxwto wtznqt': 2014.0,
'cxontgz wgqugqvnxgz': 2016.0,
'cxonvvq ntwyzgrgsxto': 2017.0,
'cxov': 2016.0,
'cxov wgzogrxavnta ohuugqn otqcxwto': 2011.0,
'cxov xzntqzvnxgzvr': 2010.0,
'cxoxmrt vruyv': 2015.0,
'cxqnhkg wgzohrnvzwj otqcxwto ucn rna': 2018.0,
'cxqnhov': 2017.0,
'cxqnhov wgzohrnxzs otqcxwto': 2015.0,
'cxqnhovugrvqxo': 2015.0,
'cxqnhvr tburgjtt': 2013.0,
'cxqnhvr tburgjtt ucn rna': 2014.0,
'cxqsxzxv ntwy': 2018.0,
'cxvovn xzw': 2017.0,
'cxvwgb': 2016.5,
'cxvwgb btaxv uqxcvnt rxbxnta': 2013.0,
'cxxnyxxojo ntwyzgrgsxto': 2018.0,
'cxyvvz ztnfgqpo rna': 2017.0,
'cxzwhrhb ogrhnxgzo rna': 2015.0,
'czr': 2014.0,
'dgbt': 2013.0,
'dgdgavj': 2017.0,
'dgqrvmo': 2016.0,
'dgqxvzn': 2015.0,
'dgqxvzn ogrhnxgzo ucn rna': 2013.0,
'djk': 2017.0,
'djk ntwyzgrgsxto xzw': 2014.0,
'djrtb fvntq ogrhnxgzo': 2014.0,
'dtmxv': 2014.0,
'dtmxv sqghu xzntqzvnxgzvr': 2014.0,
'dtqgd': 2014.0,
```

```
'dtrubgw atoxsz  ntwy': 2019.0,
'dtrxhb ntwyzgrgsxto': 2013.5,
'dtzgcho xzw': 2009.0,
'dtzgzonvwp': 2019.5,
'dtzng ojontbo ucn rna': 2015.0,
'duvzdxgz': 2014.0,
'duvzdxgz xzntqzvnxgzvr': 2013.0,
'dvbuq ogenfvqt': 2014.0,
'dvcxtzn axsxnvr ugftqta mj ntrho xzntqzvnxgzvr': 2015.0,
'dvcxtzn xzegqbvnxgz ojontbo': 2014.0,
'dwttavzwt': 2016.0,
'dxrxzd': 2012.0,
'dxrxzd xzaxv': 2012.5,
'dxrxzd xzaxv ntwyzgrgsj otqcxwto': 2014.5,
'dxvgbx xzaxv': 2014.0,
'e ztnfgqpo': 2015.0,
'egdwgzz': 2018.5,
'eggauvzav': 2014.0,
'eghqpxnto xzw': 2016.0,
'egjq': 2018.0,
'egpvrvx': 2014.0,
'egqa bgngq wgbuvzj': 2016.0,
'egqa ntwyzgrgsj otqcxwto xzaxv': 2016.0,
'egqnhqt xzegntwy': 2016.0,
'egqnxztn': 2013.5,
'egqwtugxzn': 2011.0,
'egwho ogenztn': 2014.5,
'egwho tahbvnxwo': 2013.0,
'egxft xzeg srgmvr ogrhnxgz ucn rna': 2017.0,
'ehkkj rgsxd': 2015.0,
'ehlxnoh srgmvr': 2016.0,
'ehlxnoh wgzohrnxzs': 2012.0,
'ehnhqt egwho xzegntwy': 2014.5,
'ehnhqt sqghu xzaxv': 2013.0,
'ehnhqto exqon': 2017.5,
'ehoxgz uqvwnxwto': 2016.0,
'ehoxgzwyvqno': 2015.0,
'ehqrtzwg': 2016.0,
'ehrr wqtvnxct': 2016.0,
'ehzaoxzaxv': 2016.0,
'ehzzsvst': 2016.0,
'ejrt': 2018.0,
'ejrt  tdutzot bvzvstbtzn ogenfvqt egq tzntquqxoto': 2018.0,
'ejza': 2016.0,
'ena wgbuvzxto': 2017.0,
'eoo': 2017.5,
'eqtjq tztqsj': 2009.0,
'eqtoyatop': 2014.0,
'eqtoybtzh': 2015.0,
'eqtoyfgqpo': 2015.0,
'eqtoyngygbt': 2017.0,
'eqtoytq': 2019.0,
'eqtoytq otvqwyxzs egq lgm': 2018.0,
'eqtoytq rggpxzs egq lgm': 2019.0,
'eqtoytqo': 2018.0,
'eqttrvzwt': 2014.5,
'eqttrvzwt atctrgutq': 2013.0,
'eqttrvzwtq': 2015.0,
'eqttrvzwxzs': 2013.0,
'eqttwyvqst': 2015.0,
'eqtxsyn nxstq': 2016.0,
'eqvsbv avnv ojontbo': 2018.0,
'eqvwnvr vzvrjnxwo': 2016.0,
'eqvwnvr vzvrjnxwo ucn rna': 2016.0,
'eqvzprxz ntburtngz xzctonbtzno': 2013.5,
'eqvzwgzztwn': 2016.0,
'ergf tast exzvzwxvr ogrhnxgzo ucn rna': 2010.0,
'erhnhqv atwxoxgz owxtzwto  vzvrjnxwo': 2015.0,
'erhtznsqxa rxbxnta': 2014.0,
'erjndn': 2018.0,
'erjqgmt': 2014.5,
'erjygbto': 2018.0,
'ertd': 2009.0,
'ertdnqgzxwo': 2016.0,
'ertdnqvat': 2014.0,
'ertdtqv': 2016.0,
'ertdxrgvzo': 2014.0,
'ertdxtrt': 2018.0,
'erxupvqn': 2016.0,
'etnwyq': 2014.0,
'eu ntwy owxtzwt': 2013.0,
'evhqtwxv': 2017.0,
'evmygntro': 2014.0,
```

```
'evontzvr wgbuvzj': 2016.0,
'evontzvr xzaxv': 2019.0,
'evonwgrrvm ojontbo uqxcvnt rxbxnta': 2015.0,
'evqbshxat': 2016.5,
'evqtjt  rgsxonxwo bvzvstbtzn ogenfvqt': 2017.0,
'evqtugqnvr': 2014.0,
'evrvmtrrv': 2014.0,
'evrvmtrrv wgqugqvnt otqcxwto ucn rna': 2015.0,
'evrvmtrrv xzaxv': 2015.0,
'evvogo': 2015.0,
'evvogo egga otqcxwto': 2016.0,
'evwnotn': 2016.0,
'evwnotn ojontbo': 2013.0,
'evwnotn ojontbo xzaxv ucn rna': 2017.0,
'evwtmggp': 2015.0,
'evzvnxwo xzw': 2014.5,
'exatrxnj xzctonbtzno': 2015.0,
'exatrxnj xzntqzvnxgzvr': 2017.0,
'exatrxnj zvnxgzvr exzvzwxvr': 2014.0,
'exbt': 2015.0,
'exctnqvz': 2016.0,
'exgqvzg ogenfvqt ntwyzgrgsxto ucn rna': 2017.0,
'exo': 2014.0,
'exo exatrxnj xzegqbvnxgz otqcxwto': 2010.0,
'exo srgmvr': 2015.0,
'exo srgmvr mhoxztoo ogrhnxgzo': 2014.0,
'exoagb': 2017.0,
'exooxgz rvmo': 2016.0,
'exotqc': 2014.0,
'exotqc xzaxv ucn rna': 2012.5,
'exqon avnv wgqugqvnxgz': 2010.0,
'exqon fvrpxz ntwyzgrgsxto': 2016.0,
'exqon vbtqxwvz': 2013.0,
'exqon vbtqxwvz wgqugqvnxgz': 2009.5,
'exqon vbtqxwvz xzaxv': 2015.0,
'exqonoghqwt': 2016.0,
'exqonwqjwgb': 2016.5,
'exqttjt': 2017.5,
'exqttjt xzw': 2014.0,
'exsba': 2019.0,
'exwg': 2017.0,
'exwg xzaxv ucn rna': 2017.0,
'exzdtqv': 2020.0,
'exzgqvbxw xzw': 2017.5,
'exzgxn ntwyzgrgsxto xzw': 2018.0,
'exzntwyztn': 2015.0,
'exzvdvq': 2019.0,
'exzvonqv': 2016.0,
'exzvzwxvr ogenfvqt vza ojontbo': 2014.0,
'exzxovq wgqugqvnxgz zvoavi ezoq': 2017.0,
'exzxwxnj': 2015.0,
'exzzgcvnxgz ntwy ogrhnxgzo': 2017.0,
'fggitq': 2014.0,
'fgqpo vuurxwvnxgzo wg rna': 2016.0,
'fgqpouvz': 2018.0,
'fgqptdvx': 2014.5,
'fgqpxzaxv': 2017.0,
'fgqpxzs vo eqttrvzwtq': 2015.0,
'fgqpxzs vo v eqttrvzwtq': 2011.5,
'fgqraihvzn rrw': 2017.0,
'fgqrauvj': 2018.0,
'fgrntqo prhftq': 2014.0,
'fhzatqbvz nygbuogz wgbbtqwt': 2016.0,
'fjzp': 2014.0,
'fjzp rna': 2013.0,
'fjzp rxbxnta': 2015.5,
'fogentwy': 2015.0,
'ftm atctrgutq': 2002.0,
'ftmgzxot': 2012.5,
'ftmphr ogenfvqt ucn rna': 2019.0,
'ftmtzsvst': 2017.0,
'ftmvrg': 2014.5,
'ftmzgg ntwyzgrgsxto uqxcvnt rxbxnta': 2015.0,
'fton vsxrt rvmo': 2014.0,
'ftontqz axsxnvr': 2015.5,
'ftontqz axsxnvr ovzaxop': 2019.0,
'ftontqz hzxgz': 2014.0,
'ftott': 2010.0,
'ftpvzwgat': 2013.0,
'ftrro evqsg': 2013.0,
'ftsgwgb': 2017.0,
'ftxzcton': 2016.0,
'fvayfvzx vx': 2016.0,
```

```
'fvjevxq': 2018.0,
'fvjwggrxz': 2019.0,
'fvmwg': 2016.0,
'fvont bvzvstbtzn': 2013.0,
'fvptexn': 2016.0,
'fvrbvqn': 2014.0,
'fvrbvqn rvmo': 2015.0,
'fvrbvqn srgmvr ntwy xzaxv': 2015.0,
'fvrbvqn twgbbtqwt': 2012.5,
'fxkzx xzw': 2013.0,
'fxnnjmqvxzo ogenfvqt ntwyzgrgsxto ucn rna': 2018.5,
'fxootz ntwyzgrgsj': 2015.0,
'fxootz ntwyzgrgsxto': 2015.0,
'fxootz wg': 2014.5,
'fxootz xzegntwy': 2012.5,
'fxoxntwy xzegogrhnxgzo ucn rna': 2014.0,
'fxuqg': 2018.0,
'fxuqg axsxnvr': 2016.0,
'fxuqg nhqmg': 2020.0,
'fxuqg rna': 2016.0,
'fxuqg rna  xzaxv': 2017.0,
'fxuqg rxbxnta': 2018.0,
'fxwptaqxat vactznhqgho uqxcvnt rxbxnta': 2016.0,
'fxzlxn ntwyzgrgsxto ucn rna': 2016.0,
'fxzoyhnnrt': 2015.0,
'fxzoyhnnrt ogenfvqt xzaxv ucn rna': 2015.0,
'fxzsxej': 2014.0,
'fygrtovrtmgd xzntqztn ucn rna': 2014.0,
'fyttrotjt': 2019.0,
'fyttrotjt ntwyzgrgsxto xzaxv ucn rna': 2017.0,
'fyvnexd': 2017.0,
'fyvnexdwgb': 2019.0,
'fyxntyvn lq': 2018.0,
'fyxqruggr wgqugqvnxgz': 2014.5,
'fzo srgmvr otqcxwto': 2016.0,
'g ogrhnxgzo xzw': 2016.0,
'gagg': 2018.0,
'gatoov': 2018.5,
'gatoov xzw': 2020.0,
'gbtsv ytvrnywvqt': 2016.5,
'gbxwo xzntqzvnxgzvr': 2017.0,
'gbzxfjot': 2017.0,
'gbzxt ogrhnxgzo': 2012.5,
'gdvzt uvqnztqo rxbxnta': 2016.0,
'gdxstz': 2012.0,
'gemhoxztoowgb gem ntwy ucn rna': 2014.0,
'ggarto ntwyzgrgsxto': 2016.0,
'ggarto ntwyzgrgsxto ucn rna': 2018.0,
'ghnnyxzpxzs ucn rna': 2015.0,
'gjg': 2018.0,
'gjg ntwy': 2019.0,
'gjg qggbo': 2019.0,
'gjt qxwpoyvf': 2017.0,
'gmguvj': 2018.0,
'gmltwneqgznxtq ogenfvqt geo': 2015.0,
'gmotoogqj': 2017.0,
'gmotqctvx k rvmo': 2014.0,
'gnox  gmltwn ntwyzgrgsj ogrhnxgzo xzw': 2013.5,
'gnytq': 2016.0,
'gnytqo': 2016.0,
'gou rvmo': 2015.0,
'gpnv xzw': 2013.0,
'gpwqtaxn': 2015.0,
'gqgftvrny': 2016.0,
'gqvwrt': 2015.0,
'gqvwrt geoo': 2018.0,
'gqvwrt wrgha xzeqvonqhwnhqt': 2016.0,
'gqvwrt xz exzvzwxvr otqcxwto': 2017.0,
'gqvzst': 2016.0,
'gqvzst bvznqv': 2016.0,
'gqvzst mhoxztoo otqcxwto': 2015.0,
'gqxgz ojontb xzntsqvngqo': 2017.0,
'grd sqghu': 2015.0,
'grv trtwnqxw': 2020.0,
'grv vzx ntwyzgrgsxto ucn rna': 2015.0,
'gunhb': 2018.0,
'gunhb srgmvr ogrhnxgzo': 2016.0,
'gunhbxzoxsyn xzaxv uqxcvnt rxbxnta': 2015.0,
'gunjb': 2015.0,
'gunjbjkt': 2015.5,
'guntvbxd': 2013.5,
'guntvbxd rrw': 2014.0,
'gunxbho xzegqbvnxgz xzw': 2016.5,
```

```
'gunxbvr cxqnhvr tburgjtt': 2017.0,
'gunxbxkgqj': 2018.0,
'gunxc xzw': 2017.0,
'guobd': 2013.0,
'guonqtt ogrhnxgzo': 2013.5,
'guoqvbu': 2017.0,
'guoyhm xzw': 2018.5,
'gutqvnxct': 2012.0,
'gutqvnxct oxzntwbtaxv wgbuvzj': 2011.0,
'gutqvogrhnxgzo': 2017.0,
'gutzdwtrr ntwyzgrvmo ucn rna': 2018.0,
'gutzntdn': 2016.0,
'gutzntdn ucn rna': 2019.0,
'gutzntdn xzw': 2014.0,
'guug': 2019.0,
'guug bgmxrto': 2015.0,
'guug bgmxrto xzaxv qa wtzntq': 2017.0,
'guwxng': 2016.0,
'gvnx': 2016.0,
'gw nvzztq': 2017.0,
'gwftz exzvzwxvr wgqugqvnxgz  ho': 2014.0,
'gwnqg xzw': 2018.0,
'gwnvnygqut ftm wgzohrnvzno': 2018.0,
'gxr vza zvnhqvr svo wgqugqvnxgz rna': 2017.0,
'gzbgmxrt srgmvr rxbxnta': 2015.0,
'gzoxntsg': 2017.0,
'gzsqvuy ntwyzgrgsxto rxbxnta': 2017.0,
'gzsqxa': 2016.0,
'gzt eghqny rvmo': 2020.0,
'gzt wgzctqstzwt xzw': 2017.0,
'gzt ztnfgqp tzntquqxoto': 2013.5,
'gztaxqtwn  wd bvzvstbtzn exqb': 2016.0,
'gztnqhon': 2015.0,
'gzturho': 2017.5,
'gztvooxon wgzohbtq ogrhnxgzo': 2014.5,
'gztwgb': 2015.0,
'gztwrxwp ntwyzgrgsj ucn rna': 2012.0,
'gzwtyhm': 2016.0,
'gzztn ojontbo': 2016.0,
'havvz': 2017.0,
'havvzwgb': 2016.5,
'havwxnj': 2014.0,
'hawyvrg': 2011.0,
'hguxv bgmxrt': 2011.5,
'hmgzv': 2017.0,
'hmo': 2013.0,
'hmo vs': 2012.0,
'hmtq': 2015.0,
'hmxogen': 2017.0,
'hnguxv srgmvr': 2019.0,
'hnnyhzsv ntwyzgrgsj': 2013.0,
'hnqvat ogrhnxgzo': 2018.0,
'ho tzsxzttqxzs wgbuvzj': 2019.0,
'hon srgmvr': 2014.0,
'hotqtvaj xzw': 2017.0,
'hp bxzxonqj ge atetzwt': 2015.5,
'hqlvztn': 2016.0,
'hqmvz rvaatq': 2010.5,
'hqmvz uqg': 2015.0,
'hqmvz wgbuvzj': 2019.0,
'hqmvzwrvu': 2016.0,
'hqmvzwrvu xzaxv ucn rna': 2019.0,
'hsvb': 2018.5,
'hsvb ogrhnxgzo ucn rna': 2018.0,
'hufgqp': 2017.0,
'huongd': 2014.0,
'husqvawgb': 2017.0,
'huuwr': 2015.0,
'hvzsntbvz ntwyzgrgsxto ucn rna': 2019.0,
'hxtn wyvzaxsvqy': 2020.0,
'hxuvny': 2014.0,
'hzmda xzw': 2014.0,
'hznyxzpvmrt': 2020.0,
'hznyxzpvmrt ogrhnxgzo': 2020.0,
'hzowqvbmr xzw': 2016.0,
'hzpzgfz rrw': 2012.0,
'hztbugrjta': 2017.0,
'hztburgjta': 2016.0,
'hzvwvatbj': 2018.0,
'hzxbgzx': 2016.0,
'hzxctqoxnj ge bvoovwyhotnno vbytqon': 2019.0,
'hzxctqoxnj ge xrrxzgxo vn wyxwvsg': 2018.0,
'hzxej': 2015.0,
```

```
'hzxej ntwyzgrgsxto': 2016.0,
'hzxej ntwyzgrgsxto ucn rna': 2014.0,
'hzxgz mvzp ge ofxnktqrvza': 2013.0,
'hzxnta gzrxzt xzw': 2018.0,
'hzxnta ntwyzgrgsxto': 2014.0,
'hzxnta ytvrny sqghu gunhb': 2017.0,
'hzxntaytvrny sqghu': 2015.0,
'hzxntaytvrny sqghu xzegqbvnxgz otqcxwto': 2010.0,
'hzxojo': 2015.0,
'hzxojo xzaxv': 2015.5,
'hzxrgastqo': 2018.0,
'hzxrgs': 2017.0,
'hzxrtctq': 2011.0,
'hzxwgbbtqwt': 2014.0,
'i ntwyzgrgsxto': 2014.5,
'icvzntr ogenfvqt ogrhnxgzo': 2015.0,
'ifxpwxrctq': 2014.0,
'ifxpwxrctq ogrhnxgzo': 2014.0,
'igrojo ogenfvqt ucn rna': 2015.0,
'ihgcvznxo ntwyzgrgsj': 2015.0,
'ihgcvznxo ntwyzgrgsxto': 2015.0,
'ihgnxtzn ntwyzgrgsj xzw': 2015.0,
'ihmgrt': 2017.0,
'ihmt wxztbv xzw': 2015.0,
'ihton srgmvr': 2015.0,
'ihton srgmvr tzsxzttqxzs': 2013.0,
'ihtonxgzuqg': 2015.0,
'ihtoo wgqu': 2015.0,
'ihvaqvnxw xzoxsyno ucn rna': 2016.0,
'ihvaqvzn qtoghqwt': 2015.0,
'ihvatjt otwhqxnxto ucn rna': 2017.0,
'ihvqpo ntwyzgogen': 2015.0,
'ihvrjo': 2013.0,
'ihvrjo xzw': 2011.0,
'ihvrntwy tast': 2019.0,
'ihvrwgbb': 2017.0,
'ihvrwgbb xzaxv': 2018.0,
'ihvrwgbb xzaxv ucn rna': 2011.5,
'ihvrwgbb xzw': 2018.0,
'ihvrxnj wghzwxr ge xzaxv': 2019.0,
'ihvrxnjpxgop': 2016.5,
'ihvrxnjpxgop ntwyzgrgsxto': 2015.0,
'ihvrxnton sqghu': 2015.0,
'ihvrxnton xzaxv': 2012.0,
'ihvzntrv': 2015.0,
'ihvznxuyx': 2019.0,
'ihvznxuyx vzvrjnxwo': 2018.0,
'ihvznxuyx xzw': 2018.0,
'ihxkxkk xzw': 2018.0,
'ihxpq': 2014.0,
'ihxwp ytvr': 2014.0,
'ihxwprgsxw wgqugqvnxgz': 2016.0,
'ihxwptz xzw': 2012.0,
'ihxzmvj': 2019.0,
'ihxzmvj ntwyzgrgsxto': 2020.0,
'ihxznjut ntwyzgrgsxto': 2015.0,
'ihxznjut ntwyzgrgsxto xzaxv ucn rna': 2014.0,
'ihxzzgd xzw': 2017.0,
'imhqon': 2017.0,
'ioo ntwyzgogen ucn rna': 2013.5,
'iurhb': 2014.0,
'iv xzegntwy': 2019.0,
'iv xzegntwy uqxcvnt rxbxnta': 2017.0,
'ivuxngr iv': 2015.5,
'j btaxv rvmo': 2014.0,
'jghmgnxwo': 2017.0,
'jghsgc': 2015.0,
'jghurho xzw': 2014.0,
'jgpgsvfv': 2017.0,
'jhuunc xzw': 2016.0,
'jto mvzp': 2015.0,
'jtowg oqr': 2019.0,
'jtrrgf btootzstq': 2018.0,
'jtrrgfbtootzstq': 2018.0,
'jtrrgfvx': 2019.0,
'jvbvyv bngqq ogrhnxgzo xzaxv': 2016.5,
'jvbvyv bngqq ogrhnxgzo xzaxv ucnrna': 2019.0,
'jvnqv gzrxzt': 2016.0,
'jvnqv gzrxzt ucn rna': 2014.0,
'jvnqv gzrxzt uqxcvnt rna': 2014.0,
'jvoy ntwyzgrgsxto': 2015.0,
'jvqax ogenfvqt xzaxv ucn rna': 2017.0,
'jvygg xzw': 2015.0,
```

```
'jvzatd': 2016.0,
'jvznqxpo': 2013.0,
'ke xzaxv': 2015.0,
'ke xzaxv ucn rna': 2017.0,
'kgbtznhb': 2020.0,
'kgbvng': 2016.0,
'kggbqd xzw': 2019.0,
'kggbwvq': 2015.0,
'kgrg': 2016.0,
'kguobvqn': 2018.0,
'kguutqwgb': 2015.0,
'kguygu': 2012.0,
'kguzgf': 2016.0,
'kgyg wgqu': 2015.0,
'kgyg wgqugqvnxgz': 2017.0,
'khutt': 2016.5,
'khzqgge ntwy uqxcvnt rxbxnta': 2018.0,
'kjbt': 2009.0,
'kjwho': 2015.0,
'kjzsv': 2016.0,
'ko': 2018.0,
'ko voogwxvnto': 2019.0,
'kowvrtq': 2016.5,
'ktbgog ntwyzgrgsxto': 2017.5,
'kteg': 2016.0,
'ktgbtsv': 2014.5,
'ktgnvu': 2017.0,
'ktho rtvqzxzs': 2017.0,
'ktmqv ntwyzgrgsxto': 2013.5,
'ktnftqp': 2018.0,
'ktnnvmjnt unt rna': 2017.0,
'ktnv': 2018.0,
'ktnv srgmvr': 2014.0,
'ktnv xzaxv': 2015.0,
'ktonbgztj': 2018.0,
'ktonxgn ntwyzgrgsxto': 2015.0,
'ktt tzntqnvxzbtzn tzntquqxoto rna': 2014.0,
'ktz xzeg ogrhnxgzo': 2015.5,
'ktz xzegogrhnxgzo': 2014.0,
'ktzaqxct': 2017.0,
'ktzgnx': 2017.0,
'ktzi': 2016.0,
'ktzogen otqcxwto ucn rna': 2014.5,
'ktzovq ntwyzgrgsxto': 2015.0,
'ktztexno': 2016.0,
'ktzvnxd ogrhnxgzo': 2016.0,
'kvrvzag ot': 2013.0,
'kvuq btaxv rvmo': 2015.0,
'kvuwgb ogrhnxgzo ucn rna': 2016.0,
'kxbtnqxwo': 2016.0,
'kxcvbtwgb': 2012.0,
'kxrrxgho ogrhnxgzo': 2017.0,
'kxrxzsg': 2018.0,
'kxurgvz': 2017.0,
'kxusg': 2015.0,
'kxzzgc': 2017.0,
'l u bgqsvz wyvot vza wg': 2016.0,
'lav ogenfvqt': 2013.0,
'lav ogenfvqt ucn rna': 2013.0,
'lav ogenfvqt ucn rna mvzsvrgqt': 2016.0,
'lbq xzegntwy': 2016.0,
'leqgs': 2014.0,
'lgbmvj': 2016.0,
'lgctg': 2018.0,
'lgoy ntwyzgrgsj sqghu': 2018.0,
'lgoy ogenfvqt': 2019.0,
'lgwvnv': 2015.5,
'lgyz attqt': 2015.0,
'lgyzogz wgznqgr': 2012.0,
'lgyzogz wgznqgro': 2015.0,
'lhbmgnvxr': 2018.0,
'lhon axvr rxbxnta': 2015.0,
'lhouvj': 2018.0,
'lhouvj ntwyzgrgsxto': 2018.0,
'lhszgg': 2017.0,
'lhzsrtfgqpo': 2019.0,
'lhzsrtt svbto': 2015.5,
'lhzxutq ztnfgqpo': 2015.0,
'lhzxutq ztnfgqpo xzw': 2013.0,
'lof': 2013.0,
'lp ntwyzgogen': 2017.0,
'lrn': 2013.0,
'lrr': 2015.0,
```

```
 'ltnojznytojo ucn rna': 2013.0,
 'ltoho wvrro': 2015.0,
 'lttcto tqu': 2009.0,
 'ltvcxg xzaxv ucn rna': 2019.0,
 'lu bgqsvz otqcxwto': 2009.0,
 ...}
```

In [141…
```
## Filling NaN Values with median values

df.loc[df['orgyear'].isnull(),'orgyear'] = df['company_hash'].map(dictio)
```

In [143…
```
df.isnull().sum()
```

Out[143…
```
company_hash        0
email_hash          0
orgyear             0
ctc                 0
job_position        0
ctc_updated_year    0
dtype: int64
```

In [145…
```
df['ctc_updated_year'] = df[['ctc_updated_year','orgyear']].max(axis = 1)
```

In [147…
```
## Dropping rows with unusual dates

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

In [149…
```
df
```

Out[149…

	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	Others	
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 × 6 columns

Feature Engineering

In [152…
```
## Creating new feature 'Years of Experience'

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

In [154…
```
df.head()
```

Out[154…

	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	Others	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

Creating some more new features :

Designation : Learners with a CTC above their company's department average for the same experience get Flag 1, those with an equal CTC get Flag 2, and those with a lower CTC get Flag 1.

Tier : For the company's department, learners with a CTC above the department's average get Flag 1, those with an equal CTC get Flag 2, and those with a lower CTC get Flag 1.

Class : Learners with a CTC above their company average get Flag 1, those with an equal CTC get Flag 2, and those with a lower CTC get Flag 1.

In [157…

```
## Getting the 5 point summary of CTC (mean, median, max, min, count etc) on the basis of Company, Job Posi

df_new = 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')).reset_index()
df_new
```

Out[157…

	company_hash	job_position	years_of_experience	mean_ctc	median_ctc	max_ctc	min_ctc	count_ctc
0	Others		4.0	4.500000e+05	450000.0	450000	450000	1
1	Others		6.0	4.200000e+05	420000.0	420000	420000	1
2	Others		7.0	3.500000e+05	350000.0	350000	350000	1
3	Others		14.0	7.000000e+05	700000.0	700000	700000	1
4	Others	Others	2.0	2.644000e+06	810000.0	10000000	110000	5
...	...	...	...	...	...	...	...	...
66307	zxztrtvuo	member of technical staff at nineleaps	9.0	1.200000e+06	1200000.0	1200000	1200000	1
66308	zxztrtvuo	other	5.0	4.416667e+05	450000.0	450000	400000	6
66309	zxztrtvuo	other	6.0	4.250000e+05	425000.0	450000	400000	4
66310	zxztrtvuo	software developer intern	9.0	1.200000e+06	1200000.0	1200000	1200000	1
66311	zxztrtvuo	software developer intern	10.0	1.200000e+06	1200000.0	1200000	1200000	1

66312 rows × 8 columns

In [159…

```
df_merge = df.merge(df_new, on = ['company_hash','job_position','years_of_experience'], how = 'left')
df_merge.head()
```

Out[159…

	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	Others	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 [161…

```
## Creating some flags showing learners with CTC greater than the Average of their Company's department hav

def func(ctc, mean_ctc):
    """
    Assigns a flag based on the comparison of CTC with the mean CTC.

    Returns:
    int:
        - 1 if CTC is greater than the mean CTC.
        - 2 if CTC is equal to the mean CTC.
        - 3 if CTC is less than the mean CTC.
    """
    if ctc >  mean_ctc:
        return 1
    elif mean_ctc == ctc:
        return 2
    else:
        return 3
```

In [163…

```
df_merge['tier'] = df_merge.apply(lambda x: func(x['ctc'], x['mean_ctc']), axis=1)
```

In [164…

```
df_merge.head()
```

Out[164…

	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	Others	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 [165…

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

In [169…

```
df_merge.head()
```

Out[169…

	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	Others	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 [171…
```
df_merge.rename({'tier' : 'designation'},axis = 1,inplace = True)
```

In [173…
```
df_merge.head()
```

Out[173…

	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	Others	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 [175…
```
df_new1 = df.groupby(['company_hash', 'job_position']).agg(
    mean_ctc=('ctc', 'mean')
).reset_index()
df_new1
```

Out[175…

	company_hash	job_position	mean_ctc
0	Others		4.800000e+05
1	Others	Others	3.681324e+06
2	Others	a group chat application	5.000000e+05
3	Others	abap developer	5.000000e+05
4	Others	administrative clerk	5.000000e+05
...	...	...	...
22716	zxztrtvuo	fullstack engineer	8.507273e+05
22717	zxztrtvuo	ios engineer	1.237500e+06
22718	zxztrtvuo	member of technical staff at nineleaps	1.200000e+06
22719	zxztrtvuo	other	4.350000e+05
22720	zxztrtvuo	software developer intern	1.200000e+06

22721 rows × 3 columns

In [177…
```
## Doing above analysis at Company & Job Position level.flag Class with values [1,2,3]

df_merge1 = df_merge.merge(df_new1, on = ['company_hash','job_position'], how = 'left')
df_merge1.head()
```

Out[177…

	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	Others	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 [179…
```
df_merge1['class'] = df_merge1.apply(lambda x: func(x['ctc'], x['mean_ctc']), axis=1)
```

In [180…
```
df_merge1.drop('mean_ctc',axis = 1,inplace = True)
```

In [181…
```
df_merge1.head()
```

Out[181…

	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	Others	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 [185…

```
df_new2 = df.groupby('company_hash').agg(
    mean_ctc=('ctc', 'mean')
).reset_index()
df_new2
```

Out[185…

	company_hash	mean_ctc
0	Others	2.599167e+06
1	a ntwyzgrgsxto	1.234688e+06
2	aaqxctz avnv owxtzwto vzvrjnxwo ucn rna	9.850000e+05
3	abwavnv ojontb	7.320000e+05
4	adw ntwyzgrgsj	1.234200e+06
...	...	...
3774	zxxn rna	9.014286e+05
3775	zxyxrtzn	6.887500e+05
3776	zxyxrtzn ntwyzgrgsxto	8.170000e+05
3777	zxzlvwvqn	1.739048e+06
3778	zxztrtvuo	1.172701e+06

3779 rows × 2 columns

In [187…

```
#Repeating the same analysis at the Company level

df_merge2 = df_merge1.merge(df_new2, on = ['company_hash'], how = 'left')
df_merge2.head()
```

Out[187…

	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	Others	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 [189…

```
df_merge2['tier'] = df_merge2.apply(lambda x: func(x['ctc'], x['mean_ctc']), axis=1)
```

In [191…

```
df_merge2.drop('mean_ctc',axis = 1,inplace = True)
```

In [193…

```
df_merge2.head()
```

Out[193…

	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	Others	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 [195…

```
## Top 10 employees (earning more than most of the employees in the company) – Tier 1

df_merge2[df_merge2['tier'] == 1].sort_values('ctc',ascending = False)[:11]
```

Out[195…

	company_hash	email_hash	orgyear	ctc	job_position	ctc_up
72805	Others	29a71dd13adf6d2d497571a565bb3096cf66cb46cd1ece...	2015.0	1000150000	Others	
117579	obvqnuqxdwgb	5b4bed51797140db4ed52018a979db1e34cee49e27b488...	2018.0	255555555	fullstack engineer	
3300	Others	06d231f167701592a69cdd7d5c825a0f5b30f0347a4078...	2021.0	250000000	Others	
93783	qvzaonva	fa964c3863d58f39cfa98248869a7aa41fafd4edd106cb...	2012.0	200000000	support engineer	
93784	Others	005f352591238b78024ed320789c30e7815f0eea52ed98...	2020.0	200000000	Others	
8074	zgzt	232e5b21da629a8acb3bdef7b49e57b28fc3bf64ae2687...	2023.0	200000000	engineering leadership	
21289	Others	7e447c2a4390a212cb825a72991d04251b2d943a1daf8d...	2016.0	200000000	Others	
8164	fxuqg rxbxnta	4926be010b57f52aefce6c9e2f01d43c3d4fc9e6860b25...	2017.0	200000000	support engineer	
21283	Others	dbfd099248b99c95261d19ab8f40c0b7b8aacae99d09c3...	2018.0	200000000	fullstack engineer	
101571	ytqt ntwyzgrgsxto	98a90272cbba6e6e9ca94981824f3465f3d34567cb065f...	2015.0	200000000	data analyst	
21240	Others	e8e947453901b8540be1ea26003ddaf5a27e327bdf7c8f...	2018.0	200000000	Others	

In [197…

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

df_tier1 = df_merge2[df_merge2['tier'] == 1]
df_tier_1_class1 = df_tier1[df_tier1['class'] == 1]
df_ds = df_tier_1_class1[df_tier_1_class1['job_position'] == 'data scientist']

dftop10_ds = (df_ds.groupby('company_hash')
              .apply(lambda x: x.nlargest(10, 'ctc'))
              .reset_index(drop=True))

dftop10_ds
```

Out [197…

	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	2e1d492bc09bfe0d4cc9757a9c63a296c1527af1c8ecc8...	2021.0	100000000	data scientist	
4	Others	e7722fb701c61e5cad82c39ee8bf3debe160d429b72c64...	2015.0	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 [198…

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

df_tier1 = df_merge2[df_merge2['tier'] == 1]
df_tier_1_class3 = df_tier1[df_tier1['class'] == 3]
df_ds = df_tier_1_class3[df_tier_1_class3['job_position'] == 'data scientist']

dfbottom10_ds = (df_ds.groupby('company_hash')
              .apply(lambda x: x.nsmallest(10, 'ctc'))
              .reset_index(drop=True))

dfbottom10_ds
```

Out [198…

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

322 rows × 10 columns

In [201…
```
## Bottom 10 employees (earning less than most of the employees in the company)- Tier 3

df_merge2[df_merge2['tier'] == 3].sort_values('ctc',ascending = True)[:10]
```

Out[201…

	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	Others	2
183664	Others	75357254a31f133e2d3870057922feddeba82b88056a07...	2019.0	16	Others	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 [203…
```
## Top 10 companies (based on their CTC)

df_merge2.groupby('company_hash').agg('mean','ctc').sort_values('ctc',ascending = False)[:10].reset_index()
```

Out[203…

	company_hash	orgyear	ctc	ctc_updated_year	years_of_experience	designation	class	tier
0	xzaxvmhrro	2016.285714	5.374143e+07	2020.142857	8.714286	2.000000	1.857143	2.428571
1	obvqnuqxdwgb	2017.666667	4.395259e+07	2019.166667	7.333333	2.333333	2.500000	2.666667
2	ho tzsxzttqxzs wgbuvzj	2018.714286	4.348714e+07	2021.428571	6.285714	2.000000	2.000000	2.428571
3	ouxwtltn rxbxnta	2012.200000	4.058400e+07	2020.000000	12.800000	2.000000	2.000000	2.600000
4	ovxzn sgmvxz srvoo xzaxv ucnrna	2017.200000	4.056800e+07	2020.200000	7.800000	2.000000	2.000000	2.600000
5	axowgctq agrrvq	2015.400000	4.054400e+07	2018.800000	9.600000	2.000000	2.000000	2.600000
6	exqonoghqwt	2017.000000	4.038480e+07	2020.000000	8.000000	2.000000	2.000000	2.600000
7	uyxrxuo xzaxv	2014.400000	3.770400e+07	2020.200000	10.600000	2.000000	2.200000	2.600000
8	fgqraihvzn rrw	2017.166667	3.495833e+07	2018.833333	7.833333	1.833333	1.833333	2.333333
9	egdwgzz	2017.000000	3.423250e+07	2020.166667	8.000000	2.000000	2.000000	2.666667

In [205…
```
## Top 2 positions in every company (based on their CTC)
```

In [207…
```
df_merge2.groupby(['company_hash', 'job_position'])['ctc'].mean().reset_index().sort_values(['company_hash'
```

Out[207…

	company_hash	job_position	ctc
65	Others	data entry	1.000000e+08
267	Others	telar	1.000000e+08
282	a ntwyzgrgsxto	other	3.716667e+06
279	a ntwyzgrgsxto	backend engineer	9.000000e+05
286	aaqxctz avnv owxtzwto vzvrjnxwo ucn rna	other	3.600000e+06
...	...	...	...
22696	zxyxrtzn ntwyzgrgsxto	backend engineer	9.925000e+05
22708	zxzlvwvqn	product manager	4.900000e+06
22706	zxzlvwvqn	fullstack engineer	2.181250e+06
22710	zxztrtvuo	backend architect	3.750000e+06
22713	zxztrtvuo	devops engineer	2.700000e+06

7435 rows × 3 columns

In [209…

```
## Top 10 employees in each company – X department – having 5/6/7 years of experience earning more than the

df_filered1 = df_merge2[((df_merge2['years_of_experience'] == 5) | (df_merge2['years_of_experience'] == 6)
df_filered1.groupby('company_hash')['ctc'].mean().reset_index().sort_values(by='ctc', ascending=False).grou
```

Out[209…

	company_hash	ctc
750	obvqnuqxdwgb	255555555.0
390	ho tzsxzttqxzs wgbuvzj	200000000.0
635	nqvctrnqxvzsrt	200000000.0
976	qtwpgzojo ntwy rvmo ucn rna	200000000.0
876	ovxzn sgmvxz srvoo xzaxv ucnrna	200000000.0
...	...	...
451	jvzatd	100000.0
1167	tuxw	90000.0
804	onvnt evqb	75000.0
308	ftmvrg	70000.0
1683	xzaxvzv hzxctqoxnj mrggbxzsngz	66000.0

1910 rows × 2 columns

In [211…

```
df_merge2.head()
```

Out[211…

	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	Others	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 [213…

```
df_final = df_merge2.copy()
```

In [215…

```
df_final.head()
```

Out[215…

	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	Others	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 [217…
```
df_final['years_of_experience'].max(),df_final['years_of_experience'].min()
```

Out[217… (2025.0, 0.0)

In [219…
```
df_exp = df_final['years_of_experience'].value_counts().reset_index()
df_exp
```

Out[219…

	years_of_experience	count
0	7.0	25257
1	6.0	23419
2	8.0	23225
3	9.0	23025
4	10.0	20626
...	...	...
65	1819.0	1
66	1816.0	1
67	1817.0	1
68	44.0	1
69	1825.0	1

70 rows × 2 columns

In [221…
```
## Dropping data points with more than 60 or unusual years of experience

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

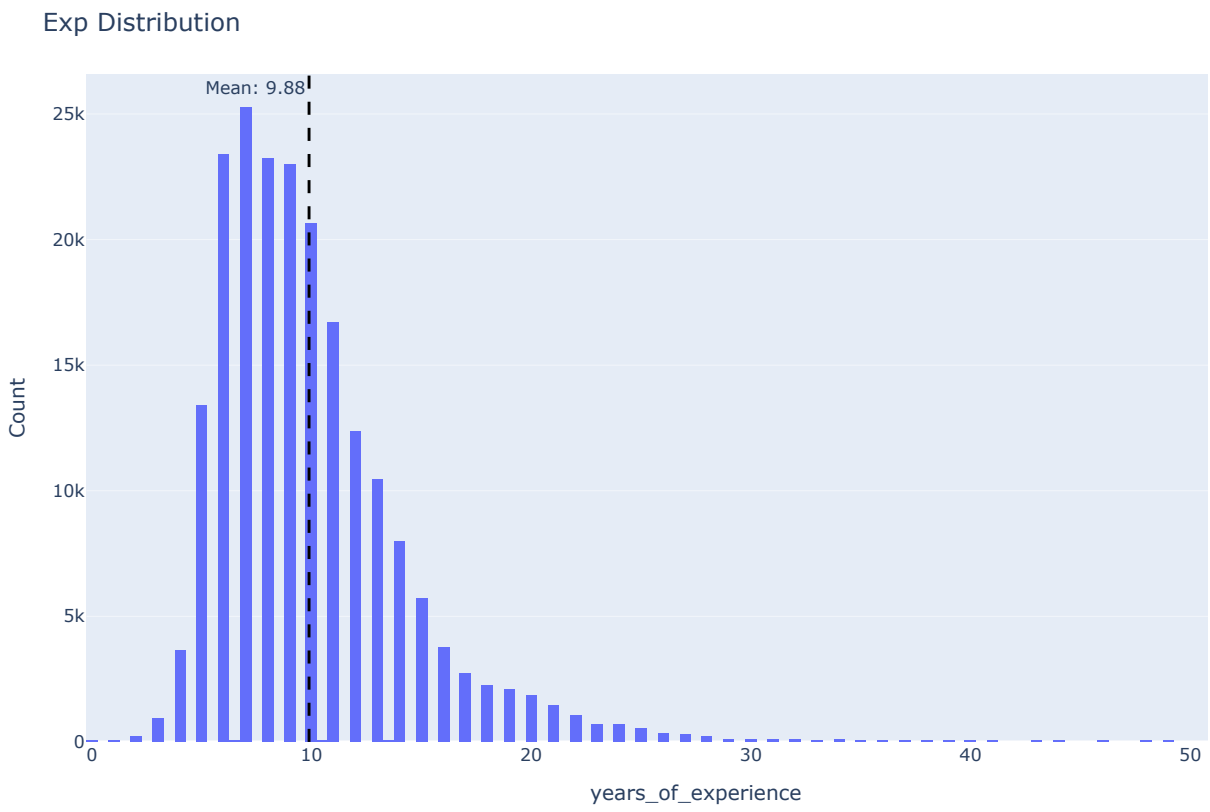
In [223…
```
df_final.head()
```

Out[223…

	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	Others	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 [225…
```
fig_2 = px.histogram(df_final, x = 'years_of_experience')
fig_2.update_layout(
    width=1000,
    height=600,
    title="Exp Distribution",
    xaxis_title="years_of_experience",
    yaxis_title="Count"
)
mean_val = df_final['years_of_experience'].mean()
```

```
median_val = df_final['years_of_experience'].median()
fig_2.add_vline(x=mean_val, line_dash="dash", line_color="black", annotation_text=f"Mean: {mean_val:.2f}",
fig_2.show()
```



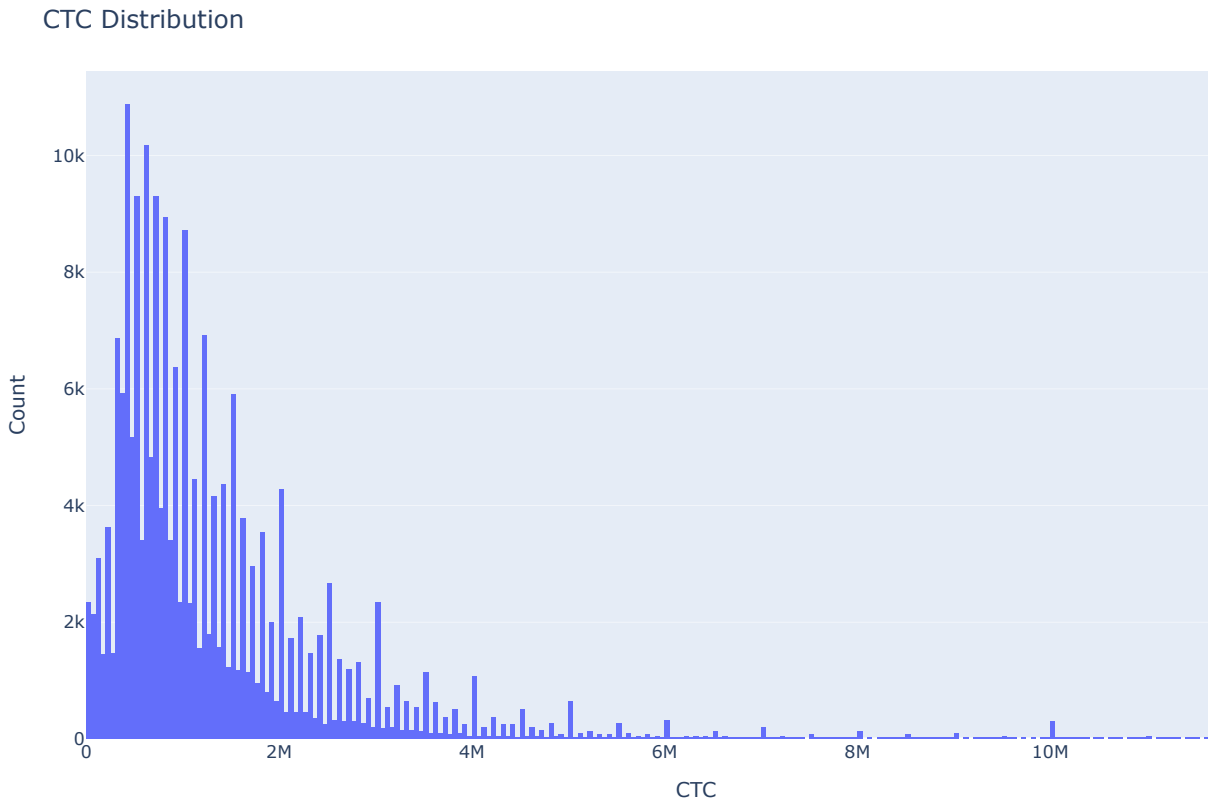
years of experience plot is right skewed with mean value of 9.88.

In [228…
```
np.percentile(df_final['ctc'],1),np.percentile(df_final['ctc'],99)
```

Out[228…
```
(37000.0, 12600000.0)
```

In [230…
```
df_final['ctc'] = df_final['ctc'].clip(lower = np.percentile(df_final['ctc'],1), upper = np.percentile(df_f
```

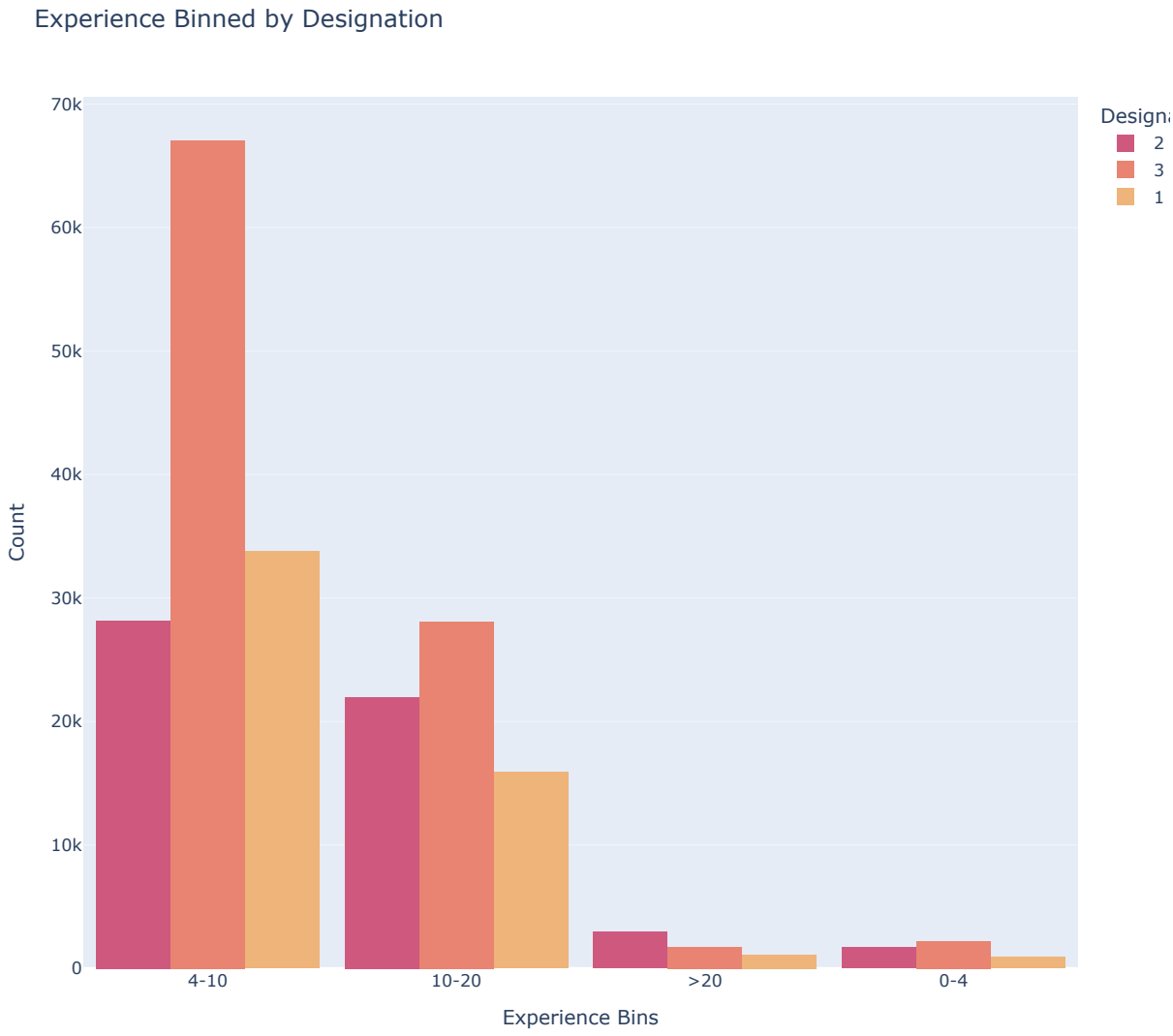
In [232…
```
fig_1 = px.histogram(df_final, x = 'ctc')

fig_1.update_layout(
    width=1000,
    height=600,
    title="CTC Distribution",
    xaxis_title="CTC",
    yaxis_title="Count"
)

fig_1.show()
```



CTC Distribution plot is heavily right skewed meaning some high values of CTC creating a long tail.

In [235…
```
df_final['orgyear'] = df_final['orgyear'].clip(lower = np.percentile(df_final['orgyear'],1), upper = np.per
```

In [236…
```
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)

fig = px.histogram(df_final,
                   x='exp_bin',
                   color='designation',
                   barmode='group',
                   title="Experience Binned by Designation",
                   color_discrete_sequence=px.colors.diverging.Temps_r
                  )

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

fig.show()
```



From the above graph we can clearly see that most employee comes under the 4-10 years of experience range with designation 3

In [240…
```
df_final.head()
```

Out[240…

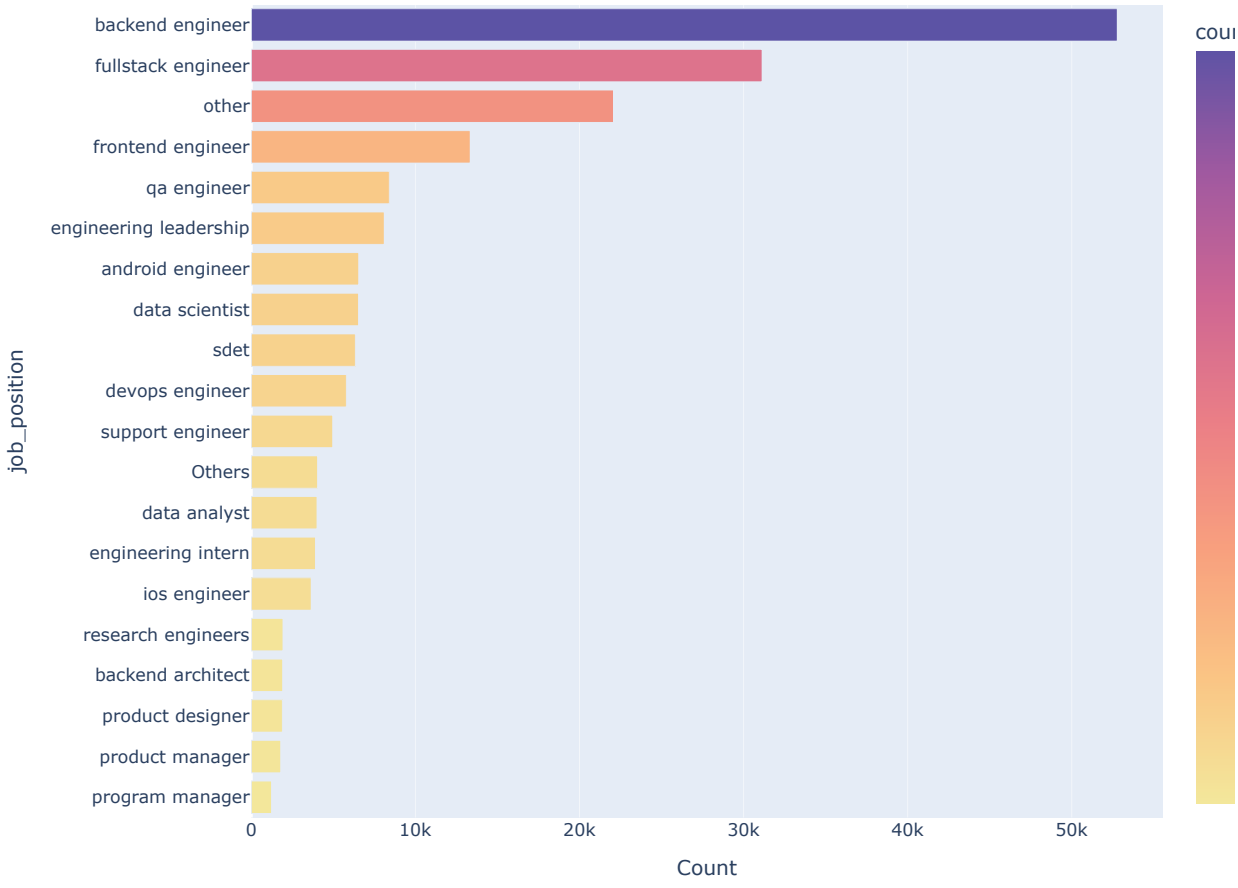
	company_hash	email_hash	orgyear	ctc	job_position	ctc_updated_ye
0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	1100000	other	2020
1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018.0	449999	fullstack engineer	2019
2	Others	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	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 [242…
```
df_final["job_position"].value_counts()
```

Out[242…
```
job_position
backend engineer                          52777
fullstack engineer                        31132
other                                     22072
frontend engineer                         13343
qa engineer                                8412
                                          ...
research assistant  software engineer         1
trainee decision scientist                    1
data specialist                               1
big data developer                            1
web  ui designer                              1
Name: count, Length: 856, dtype: int64
```

In [244…
```
job = df_final["job_position"].value_counts().reset_index().sort_values(by ='count', ascending = True).tail
```

In [246…
```
px1 = px.bar(job, x="count", y="job_position",color='count', color_continuous_scale='sunset')
px1.update_layout(
    width=900,
    height=700,
    xaxis_title="Count",
    yaxis_title="job_position",
)

px1.show()
```



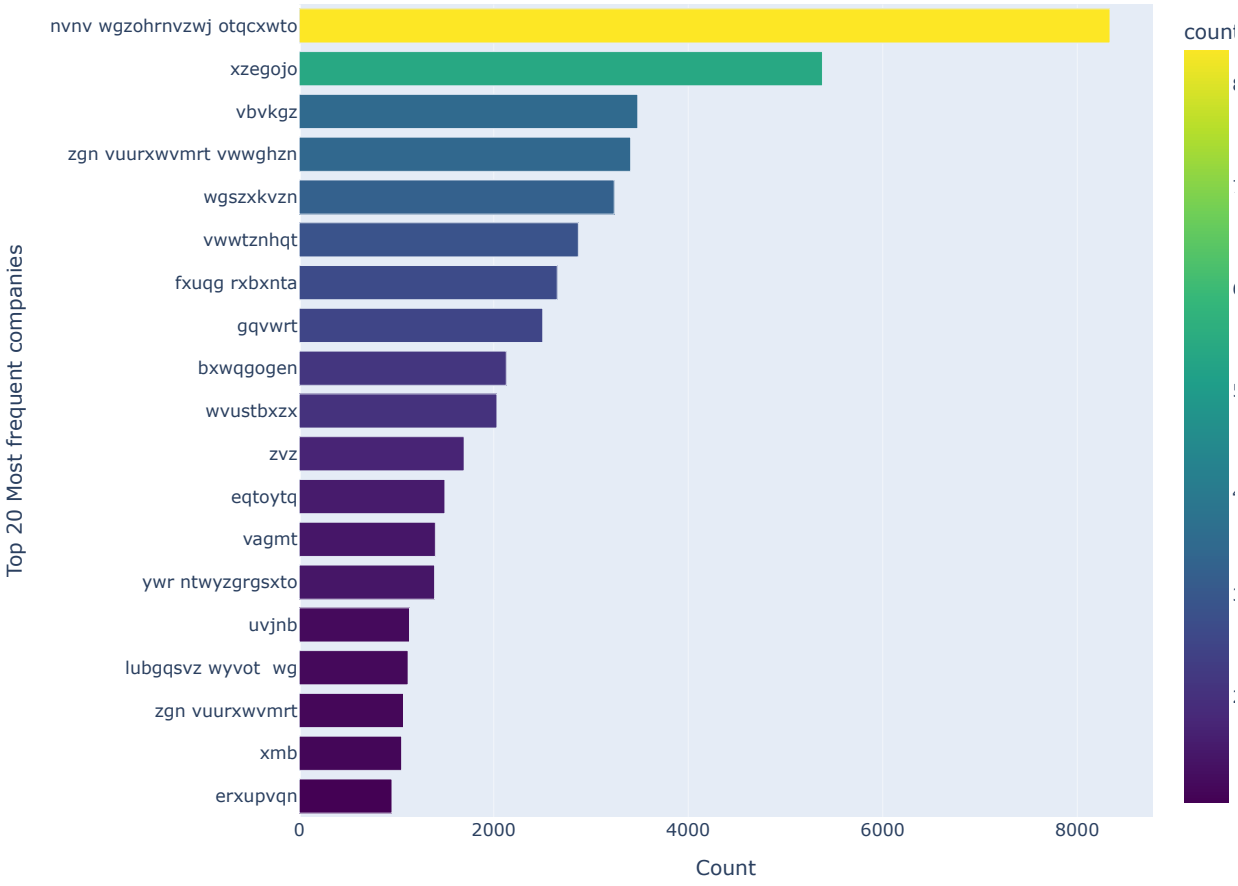
Plotted horizontal bar graph for job positions vs count.

Backend Engineer comes out at top with highest count

In [249…
```
c_hash = df_final["company_hash"].value_counts().reset_index().sort_values(by ='count', ascending = True).t
```

In [251…
```
px2 = px.bar(c_hash[~(c_hash['company_hash'] == 'Others')], x="count", y="company_hash",color='count', colo
px2.update_layout(
    width=900,
    height=700,
```

```
    xaxis_title="Count",
    yaxis_title="Top 20 Most frequent companies",
)

px2.show()
```



Plotted horizontal bar graph for company vs count.

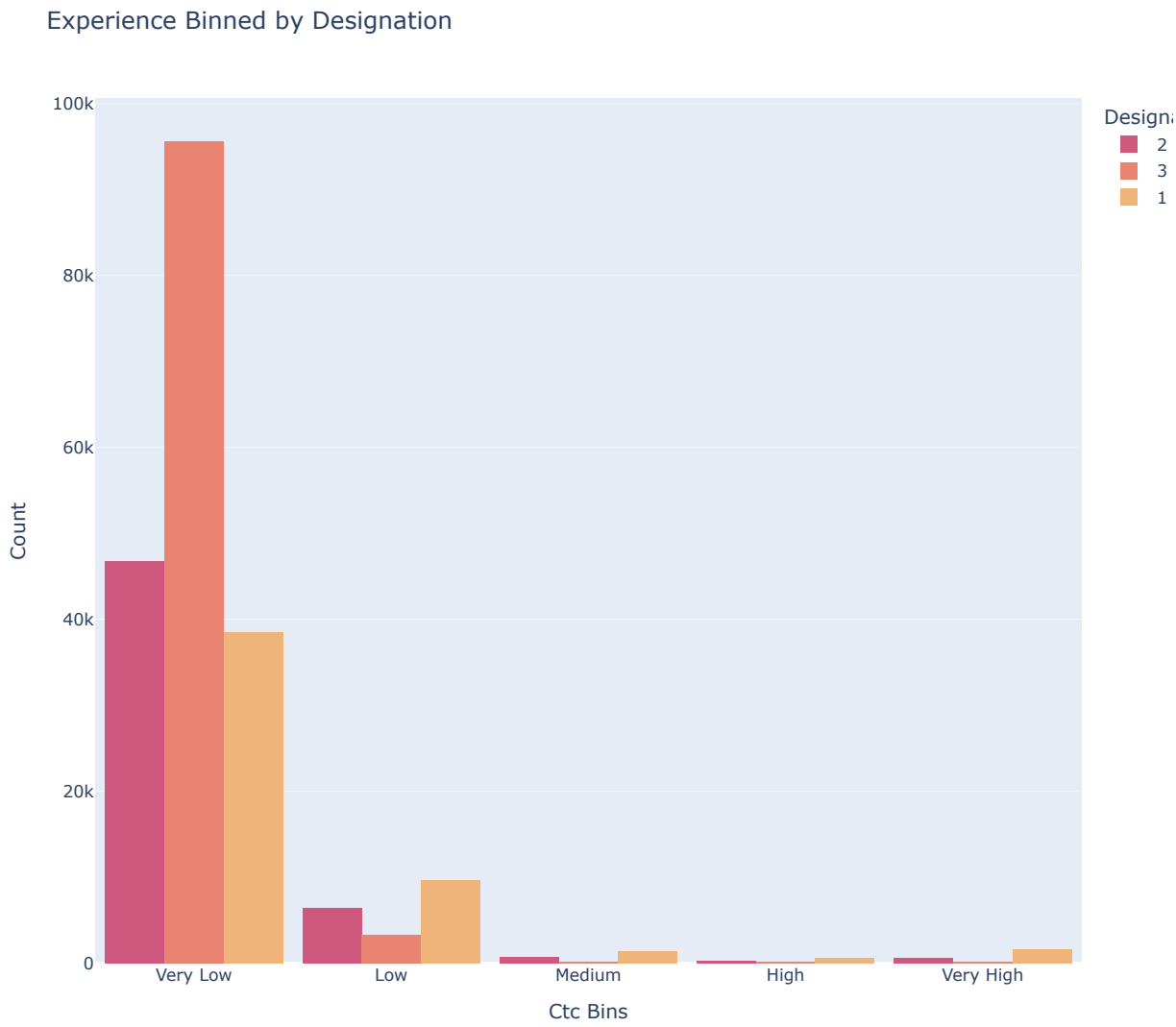
Company names are hashed

In [254…
```
df_final['ctc'].max(),df_final['ctc'].min()
```

Out[254… (12600000, 37000)

In [256…
```
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

fig3 = px.histogram(df_final,
                    x='ctc_category',
                    color='designation',
                    barmode='group',
                    title="Experience Binned by Designation",
                    color_discrete_sequence=px.colors.diverging.Temps_r
                   )

fig3.update_layout(
    width=900,
    height=800,
    xaxis_title="Ctc Bins",
    yaxis_title="Count",
    legend_title="Designation",
    bargap=0.1
)
```

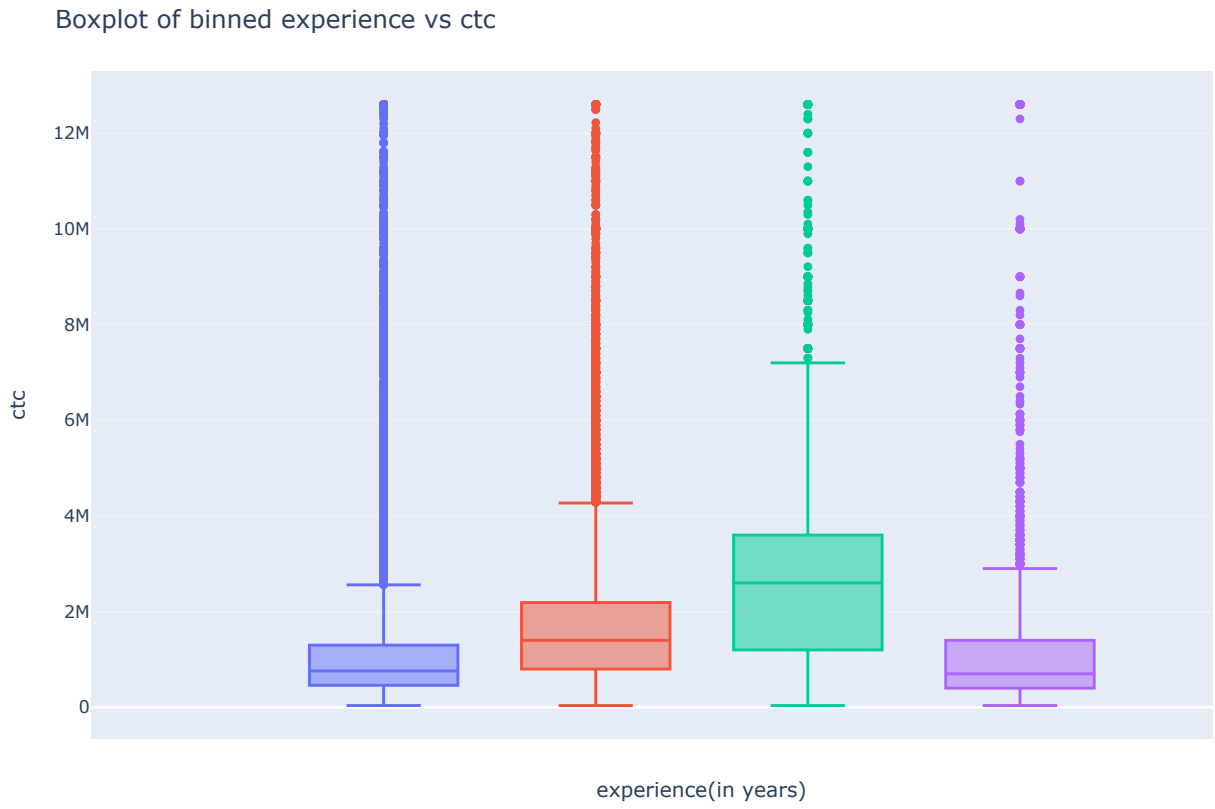
```
fig3.show()
```



We can see a high number of learners with ctc in very low range and the same pattern we have noticed in the above histogram as well

In [259…

```
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()
```

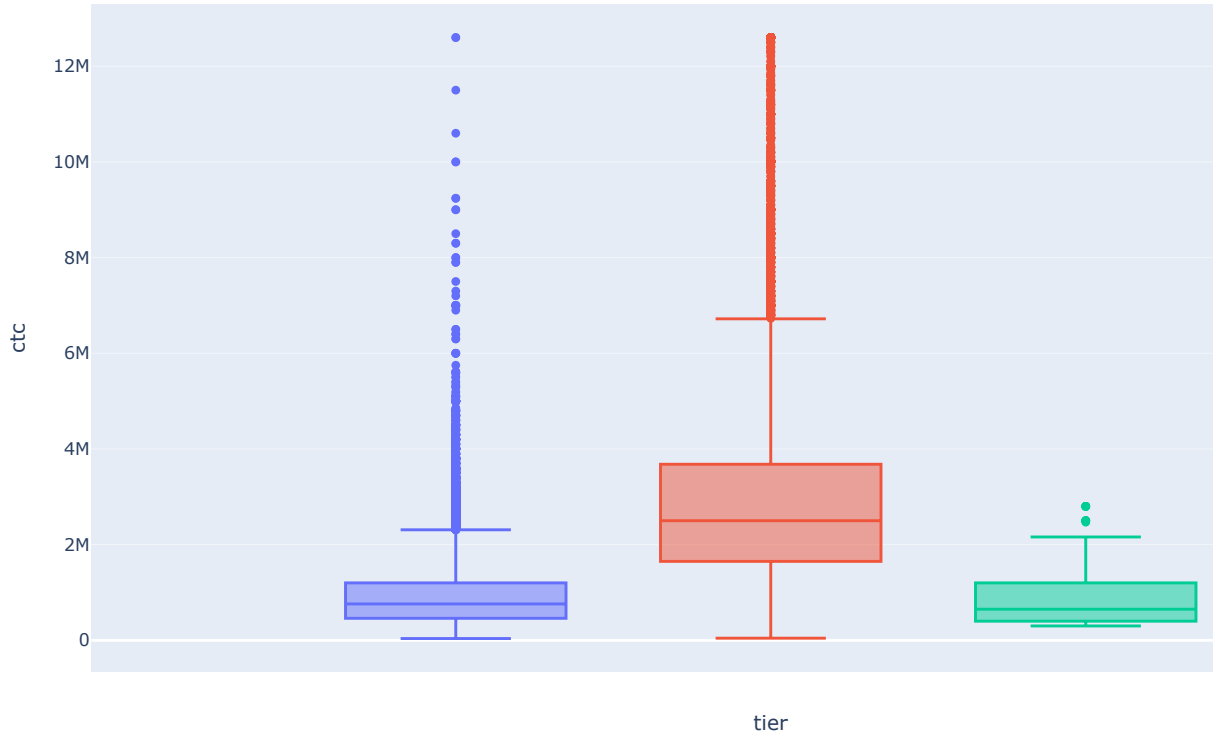


Created box plot for different years of experience bins.

Learners with more than 20 years of experience have highest range.

In [261…

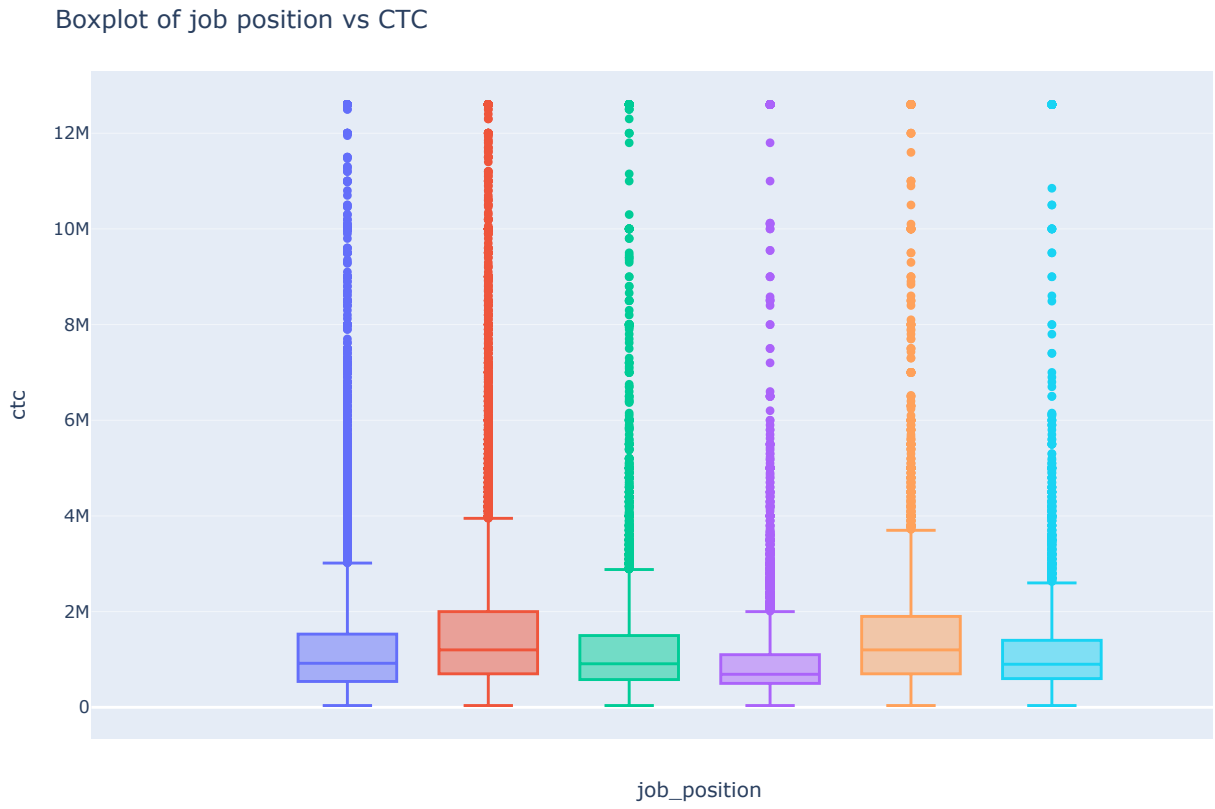
```
fig_5 = px.box(df_final, y="ctc", color = "tier")

fig_5.update_layout(
    width=1100,
    height=600,
    xaxis_title="tier",
    yaxis_title="ctc",
    legend_title="tier",
)

fig_5.show()
```



Box plot for tier 1,2 and 3

In [263…

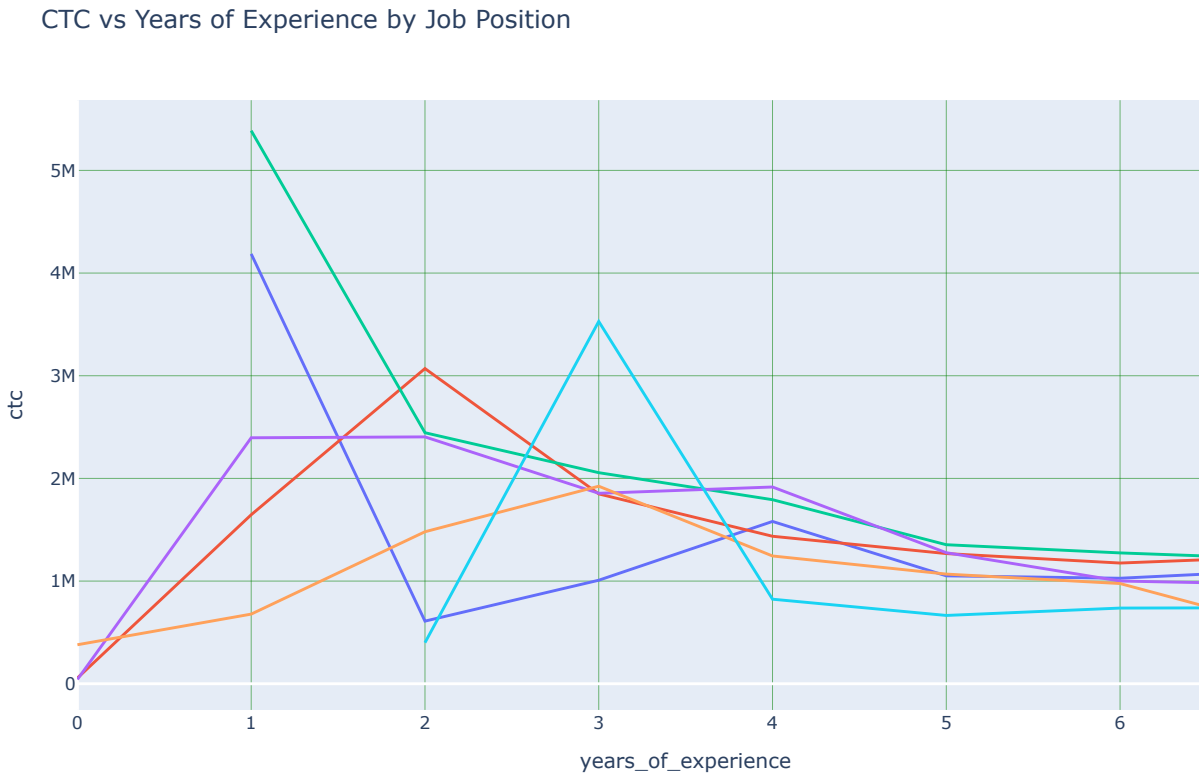
```
df_job_pos = df_final[(df_final["job_position"] == 'backend engineer') | (df_final["job_position"] == 'full

fig_5 = px.box(df_job_pos, 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()
```



Box Plot for few selected job positions with highest frequency.

In [268…

```
df_job_exp = df_final[((df_final["job_position"] == 'backend engineer') | (df_final["job_position"] == 'ful
df_grouped = df_job_exp.groupby(['job_position','years_of_experience'])['ctc'].mean().reset_index()

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.

In [271…
```
## Dropping Uncecessary Columns that do not contribute to clustering

df_final.drop(columns = {'company_hash','email_hash','orgyear','ctc_updated_year','exp_bin','ctc_category'}
```

In [273…
```
df_final.head()
```

Out[273…

	ctc	job_position	years_of_experience	designation	class	tier
0	1100000	other	9.0	2	1	3
1	449999	fullstack engineer	7.0	3	3	3
2	2000000	backend engineer	10.0	1	1	3
3	700000	backend engineer	8.0	3	3	3
4	1400000	fullstack engineer	8.0	2	1	1

In [275…
```
df_final.drop('job_position',axis = 1, inplace = True)
```

In [277…
```
df_final.shape
```

Out[277… (205623, 5)

Scaling

In [280…
```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
import scipy.cluster.hierarchy as sch
from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import DBSCAN
```

In [282…
```
scaler = StandardScaler()
```

```
df_final_scaled = scaler.fit_transform(df_final)
```

In [284…
```
df_final_scaled
```

Out[284…
```
array([[-0.18800619, -0.20854201, -0.27944008, -1.5087658 ,  0.5153169 ],
       [-0.57816162, -0.68083909,  0.93296065,  0.71210627,  0.5153169 ],
       [ 0.3522082 ,  0.02760654, -1.49184082, -1.5087658 ,  0.5153169 ],
       ...,
       [-0.42810147, -1.38928471, -1.49184082, -1.5087658 ,  0.5153169 ],
       [ 2.21294666, -0.91698763,  0.93296065,  0.71210627,  0.5153169 ],
       [-0.10397284,  0.26375508, -1.49184082, -1.5087658 ,  0.5153169 ]])
```

In [286…
```
# pip install umap-learn
```

In [288…
```
import umap
```

In [290…
```
import umap.umap_ as umap
```

In [292…
```
pca_2d = PCA(n_components = 2)
df_embed = pca_2d.fit_transform(df_final_scaled)
pca_plot = px.scatter(x = df_embed[:,0], y =df_embed[:,1])
pca_plot.update_layout(
    width=1100,
    height=600)
```

Applied Principal Component Analysis (PCA) to the entire dataset, reducing the dimensionality of all features to two principal components.

In [295…
```
tsne_2d = TSNE(n_components = 2, perplexity = 50)
df_embed_tsne = tsne_2d.fit_transform(df_final_scaled)
tsne_plot = px.scatter(x = df_embed_tsne[:,0], y = df_embed_tsne[:,1])
tsne_plot.update_layout(
    width=1100,
    height=600)
```

>

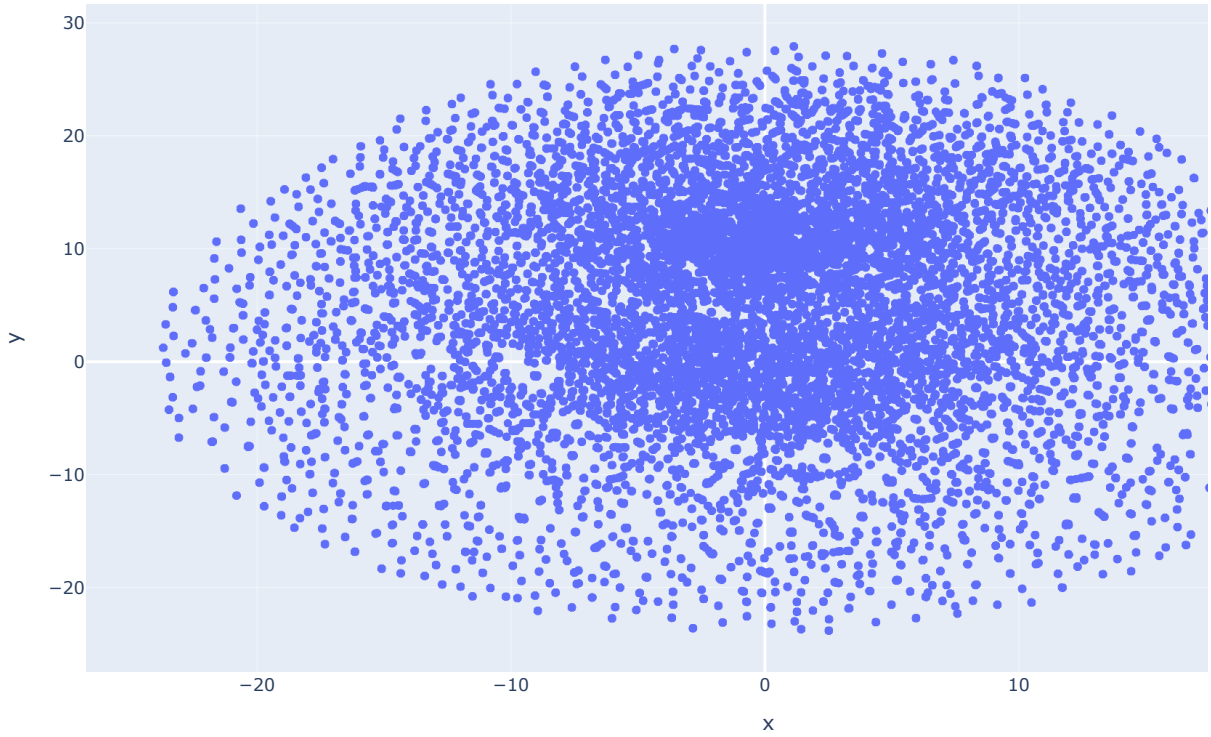
x

Applied t-Distributed Stochastic Neighbor Embedding (t-SNE) to the entire dataset with a perplexity of 50, reducing all features to a 2-dimensional representation for better visualization.

In [298…
```
umap_model = umap.UMAP(n_components=2, random_state=42)
X_umap = umap_model.fit_transform(df_final_scaled)

print(X_umap.shape)
```

```
(205623, 2)
```

In [300…
```
umap_plot = px.scatter(x = X_umap[:,0], y = X_umap[:,1])
umap_plot.update_layout(
    width=1100,
    height=600)
```

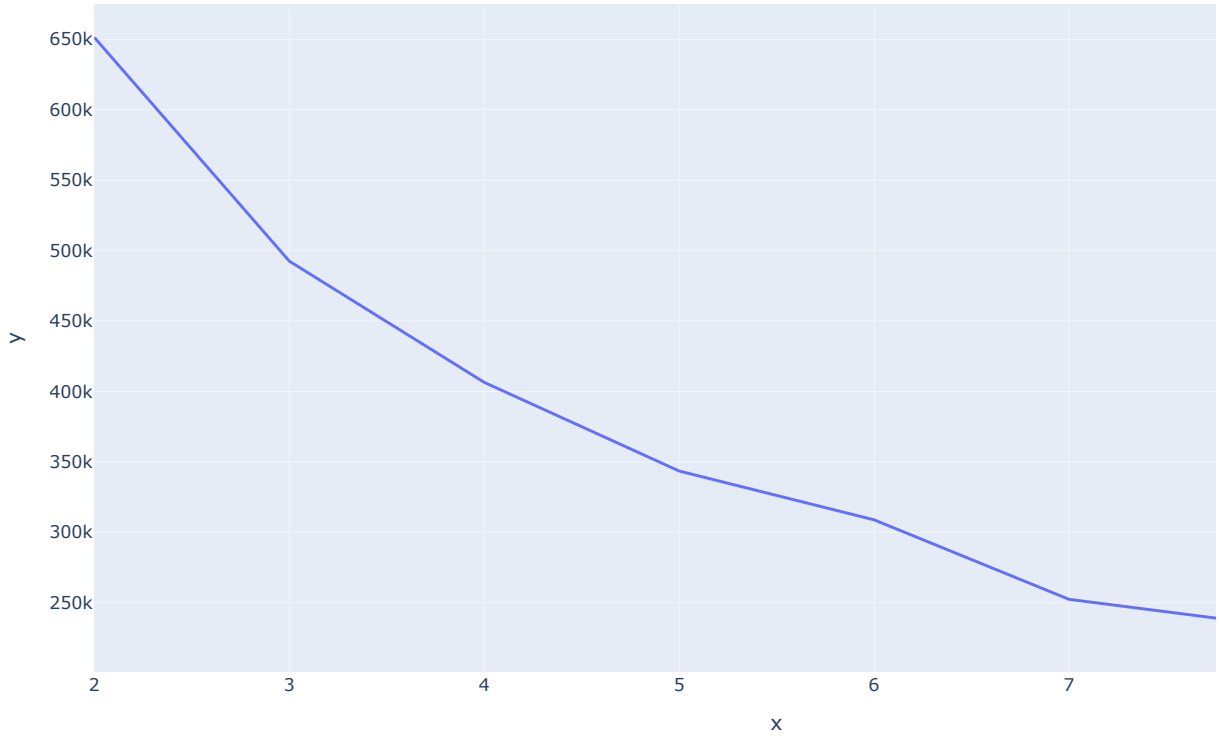


Used UMAP and reducing all features to 2-dimensional representation.

KMeans

In [305…

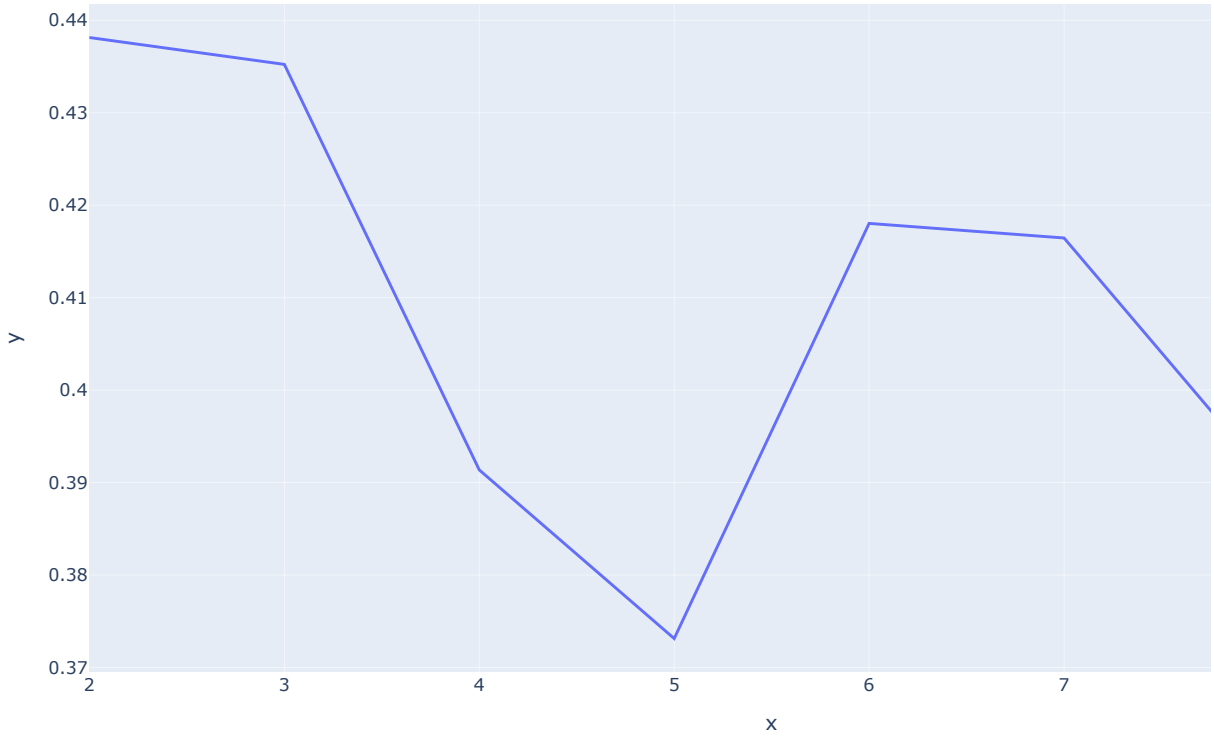
```
elbow = []
for k in range(2,10):
  kmeans = KMeans(n_clusters=k)
  kmeans.fit(df_final_scaled)
  elbow.append(kmeans.inertia_)

elbow_px = px.line(x = range(2,10), y = elbow)
elbow_px.update_layout(
    width=1100,
    height=600)
```



Employed the Elbow Method to determine the optimal value of k. However, the analysis revealed that there was no significant decrease in the Within-Cluster Sum of Squares (WCSS) for any value of k. As a result, I proceeded to explore alternative metrics to identify the best value for k.

In [309…

```
from sklearn.metrics import silhouette_score

sil_score = []
for k in range(2,10):
  kmeans = KMeans(n_clusters=k)
  kmeans.fit(df_final_scaled)
  sil_score.append(silhouette_score(df_final_scaled,kmeans.labels_))

sil_px = px.line(x = range(2,10), y = sil_score)
sil_px.update_layout(
    width=1100,
    height=600)
```

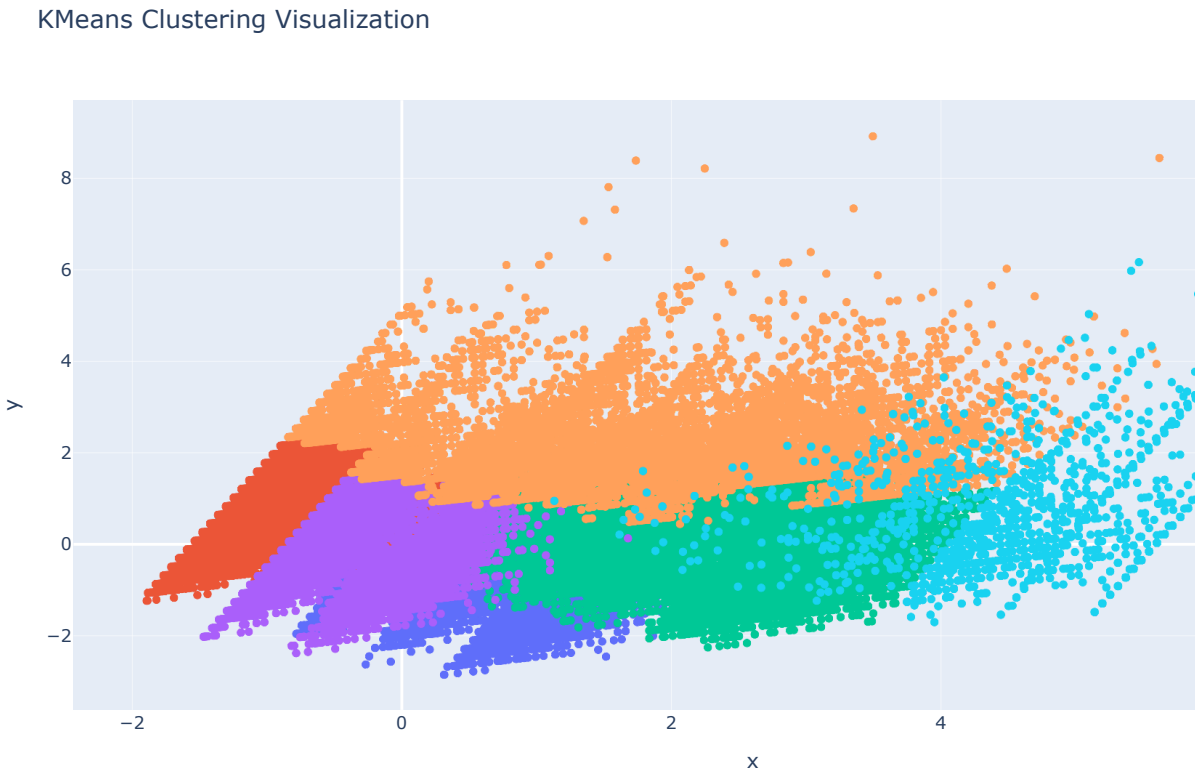


Utilized the silhouette score as an alternative metric to determine the optimal value of k. The highest score was achieved for k = 6, and the Within-Cluster Sum of Squares (WCSS) from the Elbow Method for the same value was also relatively low. Based on this analysis, we can proceed with k = 6 for model development.

In [311…
```
kmeans = KMeans(n_clusters=6)
kmeans.fit(df_final_scaled)
```

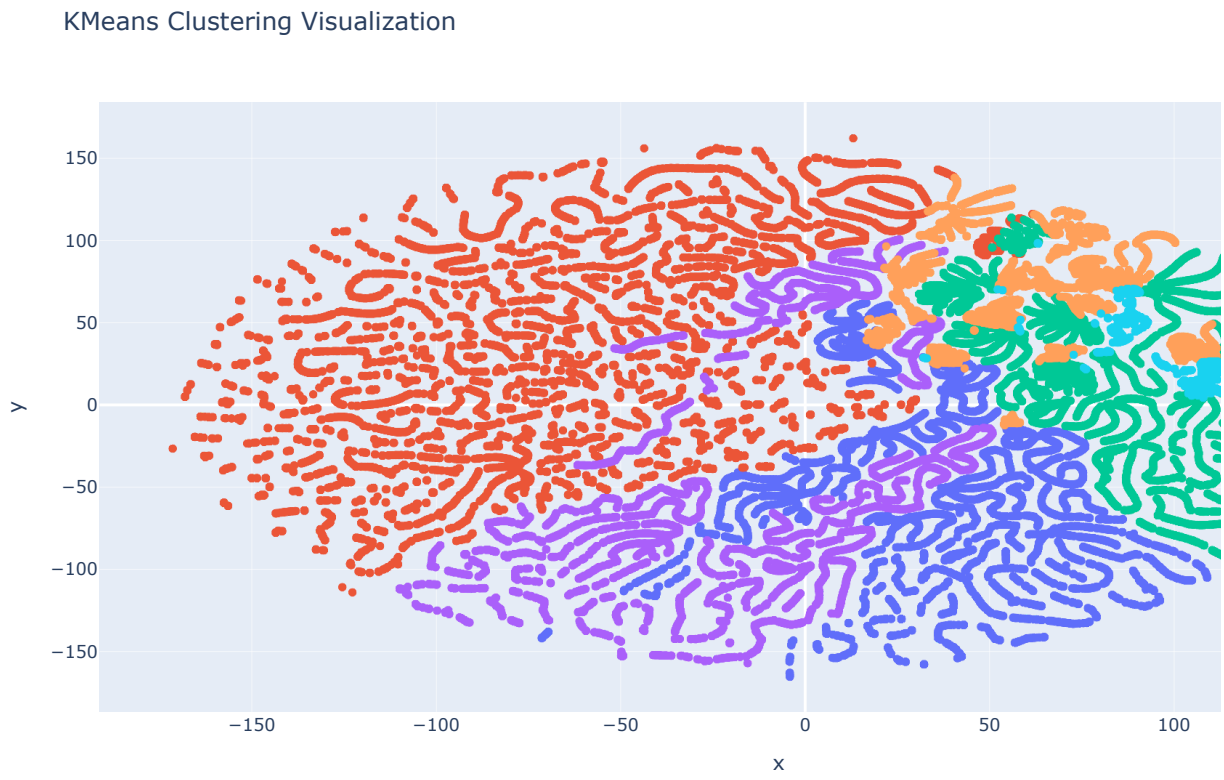
Out[311…
```
KMeans
KMeans(n_clusters=6)
```

In [313…
```
kmeans_plot = px.scatter(
    x=df_embed[:, 0],
    y=df_embed[:, 1],
    color=kmeans.labels_.astype(str),
    title="KMeans Clustering Visualization"
)

kmeans_plot.update_layout(
    width=1100,
    height=600,
    coloraxis_showscale=True
)

kmeans_plot.show()
```



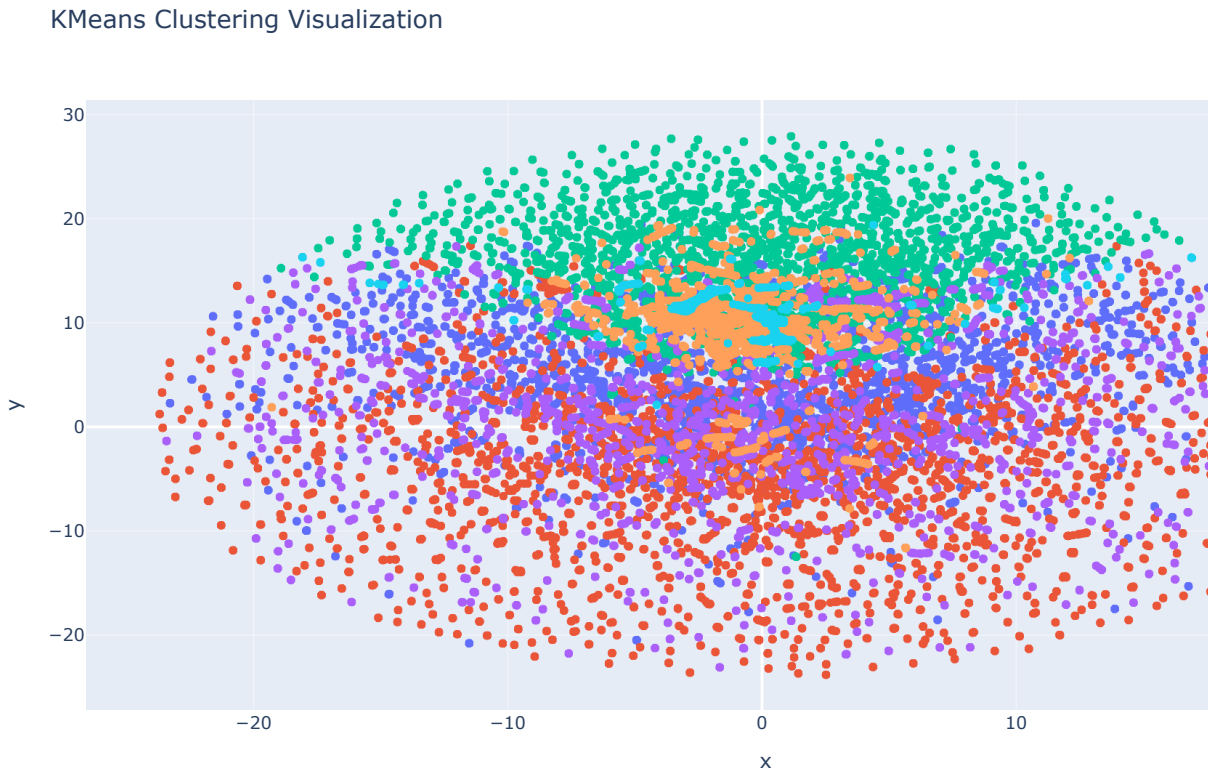
In [315…

```
kmeans_plot_tsne = px.scatter(
    x=df_embed_tsne[:, 0],
    y=df_embed_tsne[:, 1],
    color=kmeans.labels_.astype(str),
    title="KMeans Clustering Visualization"
)

kmeans_plot_tsne.update_layout(
    width=1100,
    height=600,
    coloraxis_showscale=True
)

kmeans_plot_tsne.show()
```



In [317…

```
kmeans_plot_umap = px.scatter(
x = X_umap[:,0], y = X_umap[:,1],
    color=kmeans.labels_.astype(str),
    title="KMeans Clustering Visualization"
)

kmeans_plot_umap.update_layout(
    width=1100,
    height=600,
    coloraxis_showscale=True
)

kmeans_plot_umap.show()
```



Among PCA, t-SNE, and UMAP, t-SNE provides the most effective visual representation of the clusters.

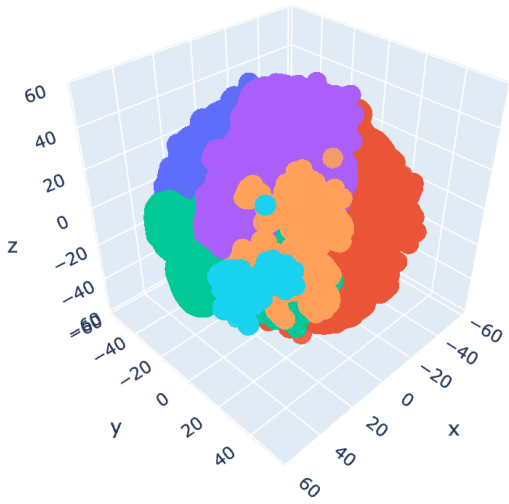
In [319…
```
tsne_3d = TSNE(n_components = 3, perplexity = 50)
df_embed_tsne_3d = tsne_3d.fit_transform(df_final_scaled)
```

In [321…
```
df_embed_tsne_3d
```

Out[321…
```
array([[ 15.950295 ,  -9.720435 ,  15.822574 ],
       [-21.27045  ,  -3.190481 ,  -3.5663397],
       [ 26.367527 , -20.947083 ,  22.991316 ],
       ...,
       [ 13.31635  , -32.08019  ,   5.652844 ],
       [  5.618409 ,  -3.704048 , -38.70838  ],
       [ 29.790348 ,  -7.976014 ,  42.163017 ]], dtype=float32)
```

In [323…
```
kmeans_plot_tsne_3d = px.scatter_3d(
    x=df_embed_tsne_3d[:, 0],
    y=df_embed_tsne_3d[:, 1],
    z=df_embed_tsne_3d[:, 2],
    color=kmeans.labels_.astype(str),
    opacity = .9,
    size_max=18,
    title="KMeans Clustering Visualization in 3D"
)

kmeans_plot_tsne_3d.update_layout(
    width=1100,
    height=600,
    # coloraxis_showscale=True
)

kmeans_plot_tsne_3d.show()
```

KMeans Clustering Visualization in 3D



Applied t-Distributed Stochastic Neighbor Embedding (t-SNE) to the entire dataset, reducing the features to three embedded components for 3D visualization of the data.

Gaussian Mixture Model

In [328…
```
gmm = GaussianMixture(n_components=4, covariance_type='full')
gmm.fit(df_final_scaled)
```

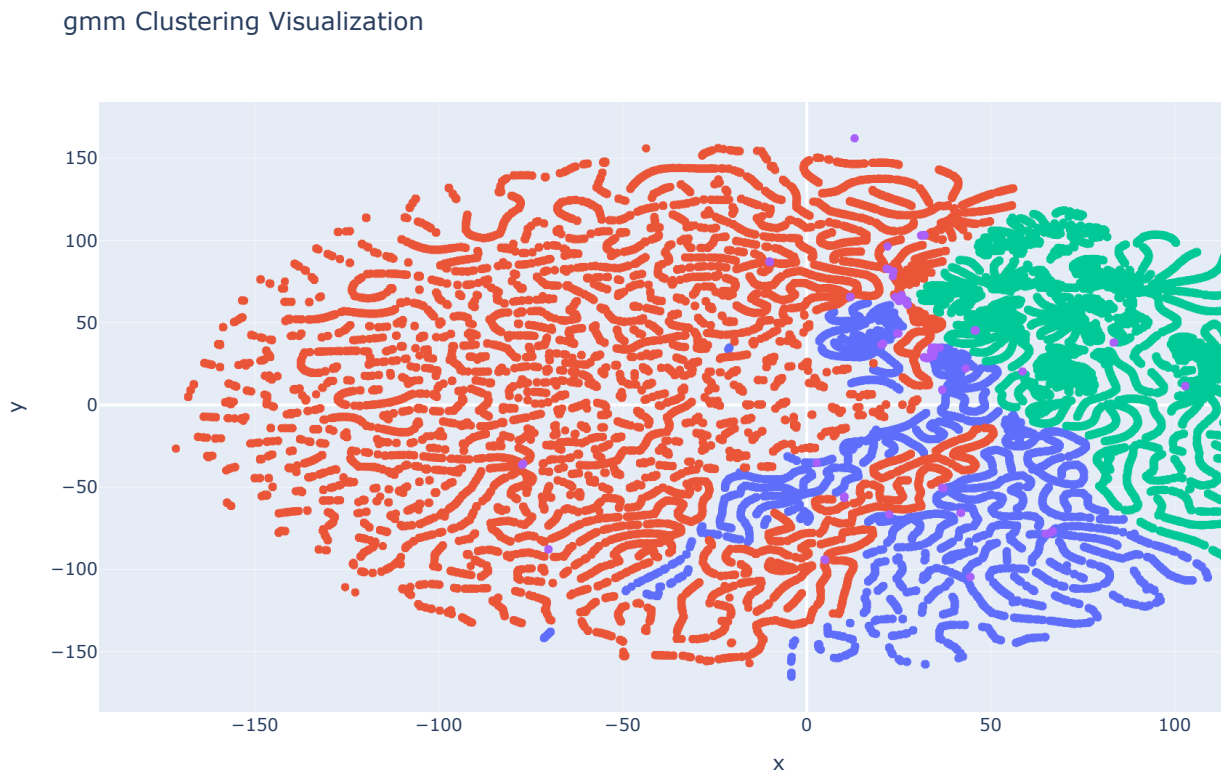
Out[328…
```
GaussianMixture(n_components=4)
```

In [329…
```
gmm_labels = gmm.predict(df_final_scaled)
```

In [332…
```
gmm_labels
```

Out[332…
```
array([0, 2, 0, ..., 0, 3, 0])
```

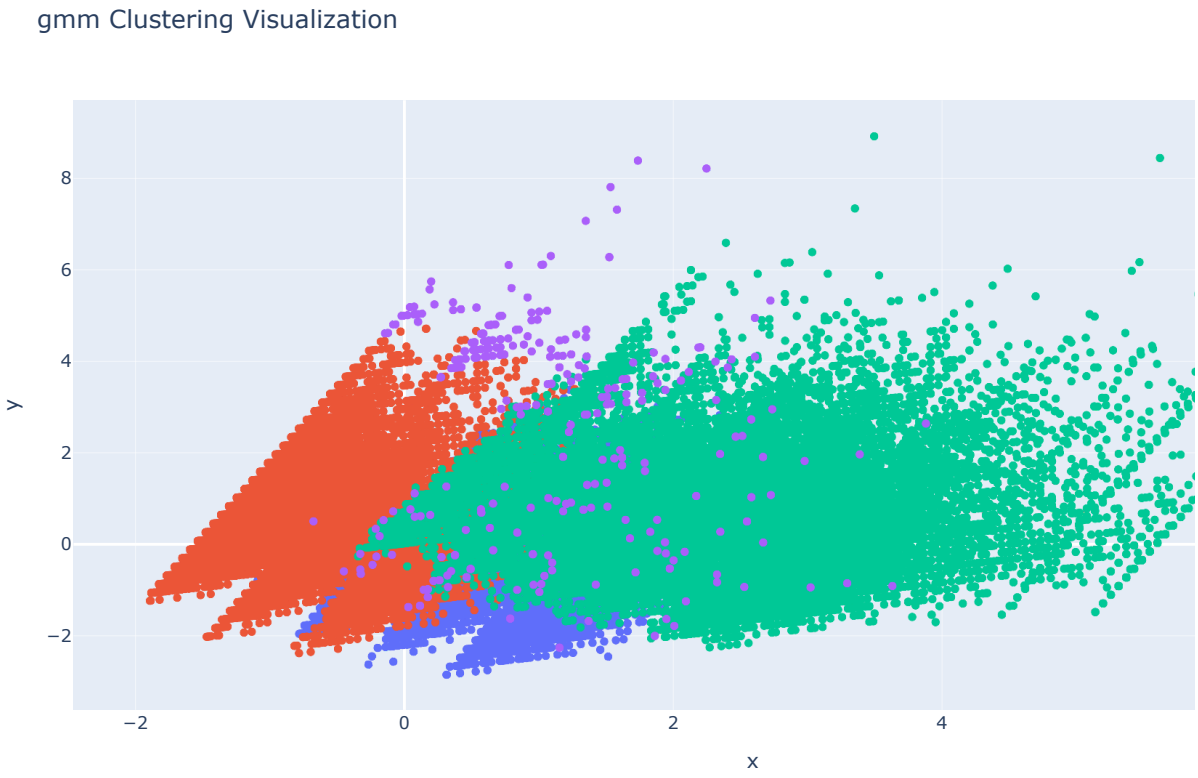
In [334…
```
gmm_plot_tsne = px.scatter(
    x=df_embed_tsne[:, 0],
    y=df_embed_tsne[:, 1],
    color=gmm_labels.astype(str),
    title="gmm Clustering Visualization"
)

gmm_plot_tsne.update_layout(
    width=1100,
    height=600,
    coloraxis_showscale=True
)

gmm_plot_tsne.show()
```



Used tSNE for better visualization of clusters in 2D

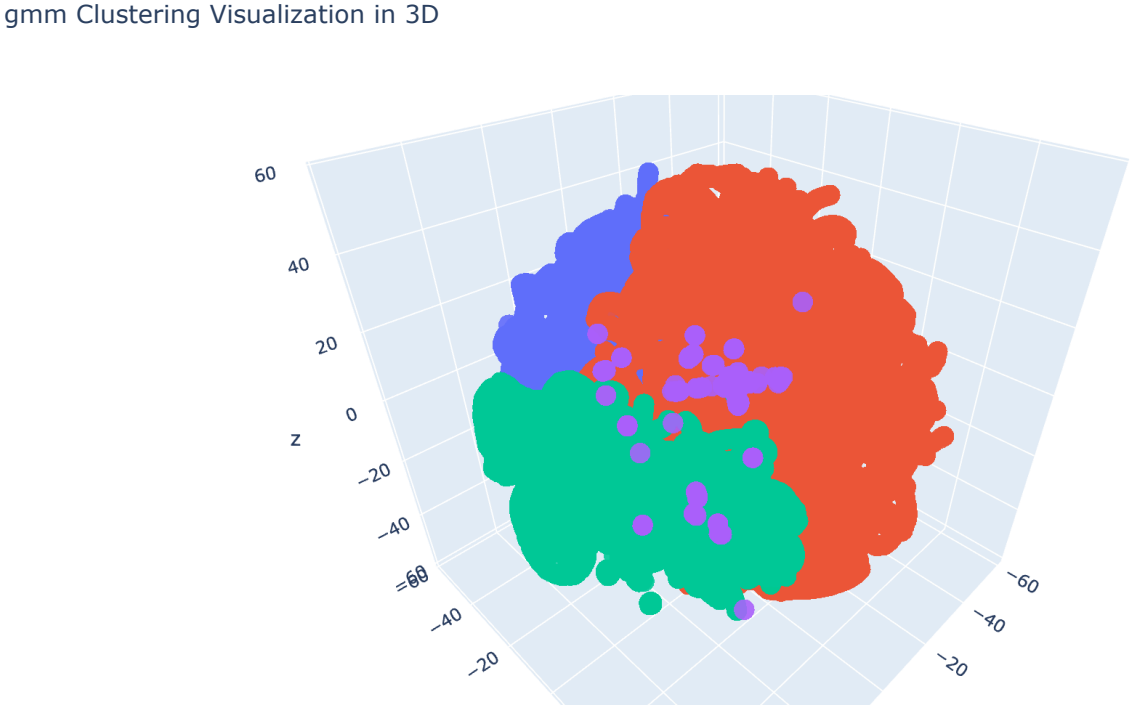
In [337…

```
gmm_plot_pca = px.scatter(
    x=df_embed[:, 0],
    y=df_embed[:, 1],
    color=gmm_labels.astype(str),
    title="gmm Clustering Visualization"
)

gmm_plot_pca.update_layout(
    width=1100,
    height=600,
    coloraxis_showscale=True
)

gmm_plot_pca.show()
```



In [339…

```
gmm_plot_tsne_3d = px.scatter_3d(
    x=df_embed_tsne_3d[:, 0],
    y=df_embed_tsne_3d[:, 1],
    z=df_embed_tsne_3d[:, 2],
    color=gmm_labels.astype(str),
    opacity = .9,
    size_max=18,
    title="gmm Clustering Visualization in 3D"
)

gmm_plot_tsne_3d.update_layout(
    width=1100,
    height=600,
)

gmm_plot_tsne_3d.show()
```



Applied t-Distributed Stochastic Neighbor Embedding (t-SNE) to the entire dataset, reducing the features to three embedded components for 3D visualization of the data.

Hierarchical Clustering

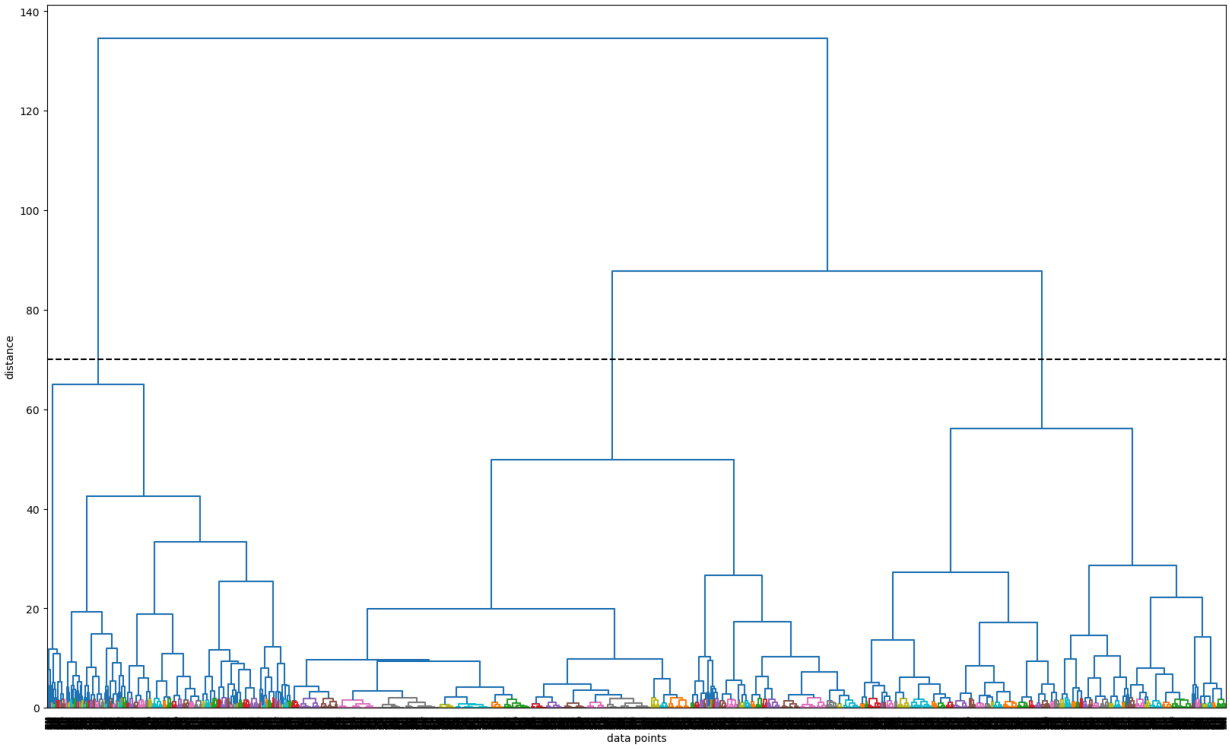
Given the large size of the dataset, we have decided to proceed with a subset of 5000 samples for hierarchical clustering analysis.

In [345…
```
df_clust = pd.DataFrame(df_final_scaled)
```

In [347…
```
h_clust = df_clust.sample(5000)
```

In [349…
```
z = sch.linkage(h_clust, method='ward')
```

In [351…
```
fig, ax = plt.subplots(figsize=(20, 12))

sch.dendrogram(z,ax=ax, color_threshold=2)


plt.xticks(rotation=90)
plt.axhline(y = 70,linestyle = '--', color = 'black')
ax.set_xlabel('data points')
ax.set_ylabel('distance')
plt.show()
```



Plotted Dendogram for the sampled data.

DBSCAN Clustering

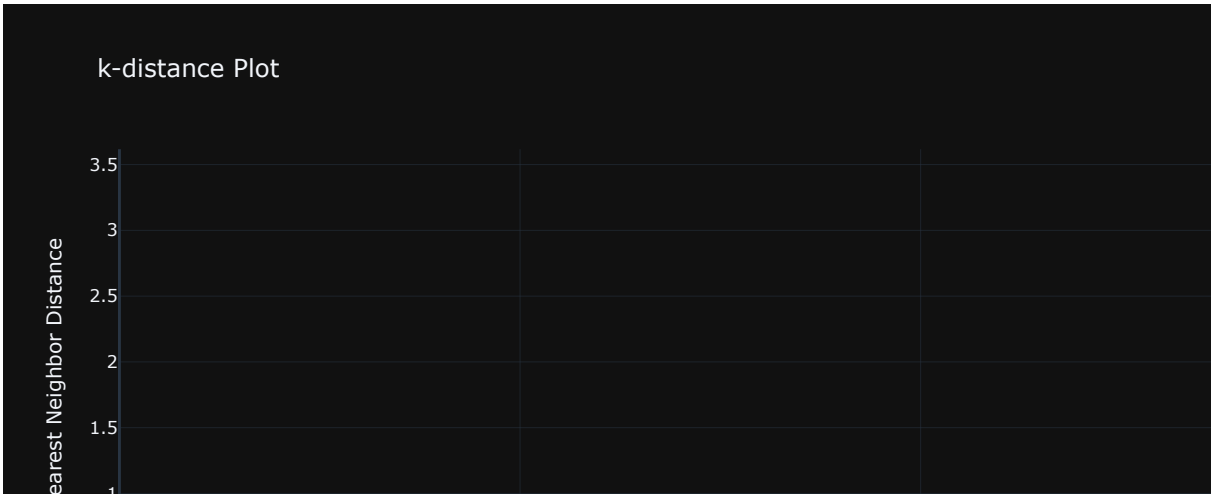
Based on the observations from the visualization, it is clear that the data exhibits varying densities, which may affect the performance of DBSCAN. However, for the purpose of this project, we have still applied DBSCAN for clustering analysis.

In [357…
```
from sklearn.neighbors import NearestNeighbors
```

Hyperparameter tuning - To find best value of epsilon, we plotted k- distance plot and observe for the best value of epsilon where it is sharply increasing or forming an elbow like structure.

In [360…
```
k = 4
neighbors = NearestNeighbors(n_neighbors=k)
neighbors.fit(df_clust)

distances, indices = neighbors.kneighbors(df_clust)

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

In [362…
```
fig = go.Figure()

fig.add_trace(go.Scatter(
    x=np.arange(len(sorted_distances)),
    y=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()
```



Best Value of epsilon comes out to be 0.25

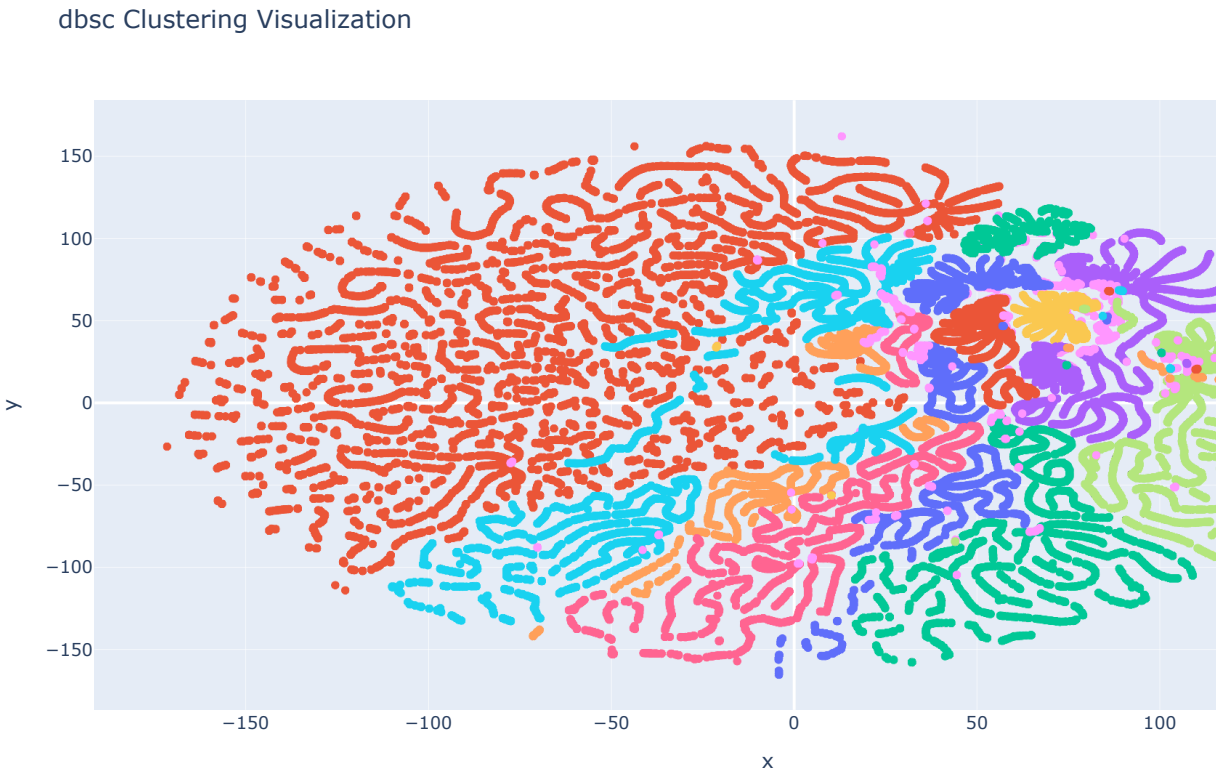
In [365…
```
dbsc = DBSCAN(eps = 0.25, min_samples = 10).fit(df_clust)
```

In [367…
```
dbsc_lbel = dbsc.labels_
```

In [369…
```
dbsc_lbel
```

Out[369…
```
array([ 0,  1,  2, ...,  2, -1,  2])
```

In [371…
```
dbsc_plot_tsne = px.scatter(
    x=df_embed_tsne[:, 0],
    y=df_embed_tsne[:, 1],
    color=dbsc_lbel.astype(str),
    title="dbsc Clustering Visualization"
)

dbsc_plot_tsne.update_layout(
    width=1100,
    height=600,
    coloraxis_showscale=True
)

dbsc_plot_tsne.show()
```



Used t-SNE to achieve a clearer 2D visualization of the clusters. Due to the varying density in the dataset, DBSCAN formed numerous clusters that do not appear to provide meaningful insights. However, for the sake of completeness, we have retained the DBSCAN clustering results.

KMeans Clustering Interpretation:

In [383…
```
kmeans.labels_
```

Out[383…
```
array([3, 0, 3, ..., 3, 0, 3], dtype=int32)
```

In [385…
```
df_clust['KMeans_labels'] = kmeans.labels_
```

In [395…
```
df_clust.columns = ['ctc', 'years_of_experience', 'designation', 'class', 'tier', 'KMeans_labels']
df_clust.head()
```

Out[395…

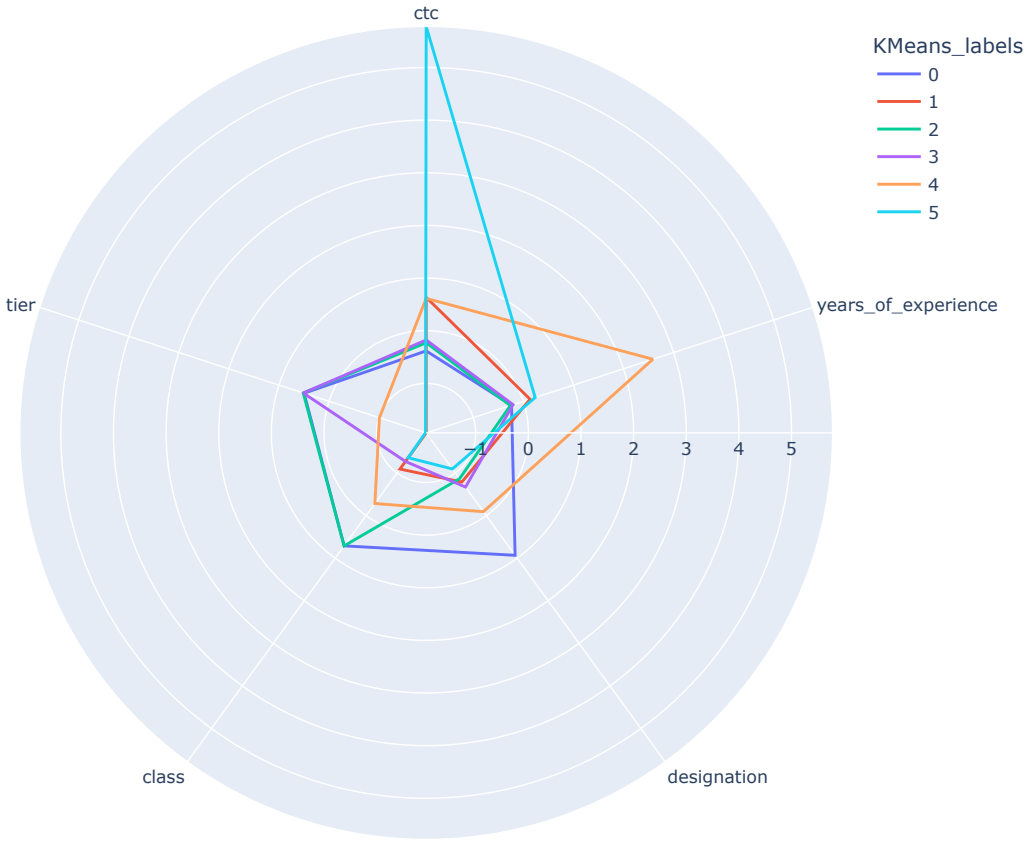
	ctc	years_of_experience	designation	class	tier	KMeans_labels
0	-0.188006	-0.208542	-0.279440	-1.508766	0.515317	3
1	-0.578162	-0.680839	0.932961	0.712106	0.515317	0
2	0.352208	0.027607	-1.491841	-1.508766	0.515317	3
3	-0.428101	-0.444691	0.932961	0.712106	0.515317	0
4	-0.007935	-0.444691	-0.279440	-1.508766	-1.941750	1

In [389…
```
polar = df_clust.groupby("KMeans_labels").mean().reset_index()
polar = pd.melt(polar, id_vars=["KMeans_labels"])
polar.head(4)
```

Out[389…

	KMeans_labels	variable	value
0	0	ctc	-0.383326
1	1	ctc	0.633402
2	2	ctc	-0.229533
3	3	ctc	-0.175508

In [391…
```
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.