Introduction

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

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

Data Dictionary:

'Unnamed 0'- Index of the dataset-

Email_hash- Anonymised Personal Identifiable Information (PII)

Company_hash- 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)

Concept Used:

Manual Clustering Unsupervised Clustering - K- means, Hierarchical Clustering

In [13]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
import numpy as np # linear algebra
```

```
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import os
import re
import seaborn as sns
from matplotlib import pyplot as plt
```

In [14]:
```
#Reading csv data
data = pd.read_csv('scaler_clustering.csv')
```

In [15]:
```
print("Dimensions of dataset ",data.shape)

data.head()
```

```
Dimensions of dataset  (205843, 7)
```

Out[15]:

	Unnamed: 0	company_hash	email_hash	orgyear	ctc	job_position	ctc_updated_year
0	0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	1100000	Other	2020.0
1	1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018.0	449999	FullStack Engineer	2019.0
2	2	ojzwnvwnxw vx	4860c670bcd48fb96c02a4b0ae3608ae6fdd98176112e9...	2015.0	2000000	Backend Engineer	2020.0
3	3	ngpgutaxv	effdede7a2e7c2af664c8a31d9346385016128d66bbc58...	2017.0	700000	Backend Engineer	2019.0
4	4	qxen sqghu	6ff54e709262f55cb999a1c1db8436cb2055d8f79ab520...	2017.0	1400000	FullStack Engineer	2019.0

In [16]:
```
data.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      153281 non-null  object 
 6   ctc_updated_year  205843 non-null  float64
dtypes: float64(2), int64(2), object(3)
memory usage: 11.0+ MB
```

Checking for Null Values in dataset

In [17]:
```
data.isna().sum()
```

Out[17]:
```
Unnamed: 0             0
company_hash          44
email_hash             0
orgyear               86
ctc                    0
job_position       52562
ctc_updated_year       0
dtype: int64
```

Data contains null values in 3 columns [company_hash, orgyear, job_position].

Checking for Duplicate Rows in dataset

In [18]:
```
len(data[data.duplicated()])
```

Out[18]:
```
0
```

Data doesn't contain any duplicate rows.

Data Preprocessing

Checking duplicated PII ids in column email_hash

In [19]:
```
data['email_hash'].value_counts().head(10)
```

Out[19]:
```
bbace3cc586400bbc65765bc6a16b77d8913836cfc98b77c05488f02f5714a4b    10
6842660273f70e9aa239026ba33bfe82275d6ab0d20124021b952b5bc3d07e6c     9
298528ce3160cc761e4dc37a07337ee2e0589df251d73645aae209b010210eee     9
3e5e49daa5527a6d5a33599b238bf9bf31e85b9efa9a94f1c88c5e15a6f31378     9
b4d5afa09bec8689017d8b29701b80d664ca37b83cb883376b2e95191320da66     8
faf40195f8c58d5c7edc758cc725a762d51920da996410b80ac4a4d85c803da0     8
4818edfd67ed8563dde5d083306485d91d19f4f1c95d193a1700e79dd245b75c     8
c0eb129061675da412b0deb15871dd06ef0d7cd86eb5f7e8cc6a20b0d1938183     8
d598d6f1fb21b45593c2afc1c2f76ae9f4cb7167156cdf93246d4192a89d8065     8
d15041f58bb01c8ee29f72e33b136e26bc32f3169a40b53d75fe7ae9cbb9a551     8
Name: email_hash, dtype: int64
```

In [20]:
```
display(data[data['email_hash'] == 'bbace3cc586400bbc65765bc6a16b77d8913836cfc98b77c05488f02f5714a4b'])
display(data[data['email_hash'] == '6842660273f70e9aa239026ba33bfe82275d6ab0d20124021b952b5bc3d07e6c'])
```

	Unnamed: 0	company_hash	email_hash	orgyear	ctc	job_position	ctc_updated_year
24109	24129	oxej ntwyzgrgsxto rxbxnta	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	2018.0	720000	NaN	2020.0
45984	46038	oxej ntwyzgrgsxto rxbxnta	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	2018.0	720000	Support Engineer	2020.0
72315	72415	oxej ntwyzgrgsxto rxbxnta	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	2018.0	720000	Other	2020.0
102915	103145	oxej ntwyzgrgsxto rxbxnta	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	2018.0	720000	FullStack Engineer	2020.0
117764	118076	oxej ntwyzgrgsxto rxbxnta	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	2018.0	720000	Data Analyst	2020.0
121483	121825	oxej ntwyzgrgsxto rxbxnta	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	2018.0	660000	Other	2019.0
124476	124840	oxej ntwyzgrgsxto rxbxnta	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	2018.0	660000	Support Engineer	2019.0
144479	145021	oxej ntwyzgrgsxto rxbxnta	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	2018.0	660000	FullStack Engineer	2019.0
152801	153402	oxej ntwyzgrgsxto rxbxnta	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	2018.0	660000	Devops Engineer	2019.0
159835	160472	oxej ntwyzgrgsxto rxbxnta	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	2018.0	660000	NaN	2019.0

	Unnamed: 0	company_hash	email_hash	orgyear	ctc	job_position	ctc_updated_year
9857	9859	ihvrwgbb	6842660273f70e9aa239026ba33bfe82275d6ab0d20124...	2017.0	2400000	QA Engineer	2020.0
10002	10006	ihvrwgbb	6842660273f70e9aa239026ba33bfe82275d6ab0d20124...	2017.0	2400000	Devops Engineer	2020.0
10583	10587	ihvrwgbb	6842660273f70e9aa239026ba33bfe82275d6ab0d20124...	2017.0	2400000	Backend Engineer	2020.0
12784	12793	ihvrwgbb	6842660273f70e9aa239026ba33bfe82275d6ab0d20124...	2017.0	2400000	Other	2020.0
20715	20729	ihvrwgbb	6842660273f70e9aa239026ba33bfe82275d6ab0d20124...	2017.0	2400000	SDET	2020.0
138253	138731	ihvrwgbb	6842660273f70e9aa239026ba33bfe82275d6ab0d20124...	2017.0	2000000	SDET	2020.0
159251	159887	ihvrwgbb	6842660273f70e9aa239026ba33bfe82275d6ab0d20124...	2017.0	2000000	Devops Engineer	2020.0
165343	166040	ihvrwgbb	6842660273f70e9aa239026ba33bfe82275d6ab0d20124...	2017.0	2000000	Other	2020.0
178749	179593	ihvrwgbb	6842660273f70e9aa239026ba33bfe82275d6ab0d20124...	2017.0	2000000	Backend Engineer	2020.0

Apparently for single Anonymised Personal Identifiable Information (PII) id there exists multiple rows with same joining dates and company but different job positions, this couldn't be possible.
We will take the first row in case of duplicated PII ids.

In [21]:
```
data = data.groupby('email_hash').first().reset_index()
```

Creating null value indicator columns (Feature Engineering)

In [22]:
```
for i in ['orgyear','ctc_updated_year','company_hash','job_position']:
    data[i+'_na'] = data[i].isna()
```

Cleaning text columns

In [23]:
```
text_cols = ['company_hash']
for i in text_cols:
    data[i] = data[i].fillna('Not Available').apply(lambda x: re.sub('[^A-Za-z0-9 ]+', '', str(x).title()))
```

Creating new columns (Feature Engineering)

In [24]:
```
data['YoE'] = data['ctc_updated_year'] - data['orgyear']
```

Frequency mean encoding

In [25]:
```
feat = 'company_hash'
data[feat] = data[feat].fillna('na')
enc_nom = (data.groupby(feat).size()) / len(data)
data[feat+'_encode'] = data[feat].apply(lambda x : enc_nom[x])

feat = 'job_position'
data[feat] = data[feat].fillna('na')
enc_nom = (data.groupby(feat).size()) / len(data)*10000
data[feat+'_encode'] = data[feat].apply(lambda x : enc_nom[x])
```

Reemoving Outliers from Orgyear column

In [26]:
```
sorted(data['orgyear'].fillna(0).astype(int).unique())
```

Out[26]: [0,
1,
2,
3,
4,
5,
6,
38,
83,
91,
200,
201,
206,
208,
209,
1900,
1970,
1971,
1972,
1973,
1976,
1977,
1979,
1981,
1982,
1984,
1985,
1986,
1987,
1988,
1989,
1990,
1991,
1992,
1993,
1994,
1995,
1996,
1997,
1998,

```
1999,
2000,
2001,
2002,
2003,
2004,
2005,
2006,
2007,
2008,
2009,
2010,
2011,
2012,
2013,
2014,
2015,
2016,
2017,
2018,
2019,
2020,
2021,
2022,
2023,
2024,
2025,
2026,
2027,
2028,
2029,
2031,
2101,
2106,
2107,
2204,
20165]
```

Removing future years, as this case is impossible to happen, also removing single digit years.

In [27]:
```
data = data[~data['orgyear'].isin([0,
 1,
 2,
 3,
 4,
 5,
 6,
 38,
 83,
 91,
 200,
 201,
 206,
 208,
 209,
 1900, 2023,
 2024,
 2025,
 2026,
 2027,
 2028,
 2029,
 2031,
 2101,
 2106,
 2107,
 2204,
 20165])]
```

In [28]:
```
data = data[~(data['YoE']<0)]
```

Exploratory Data Analysis

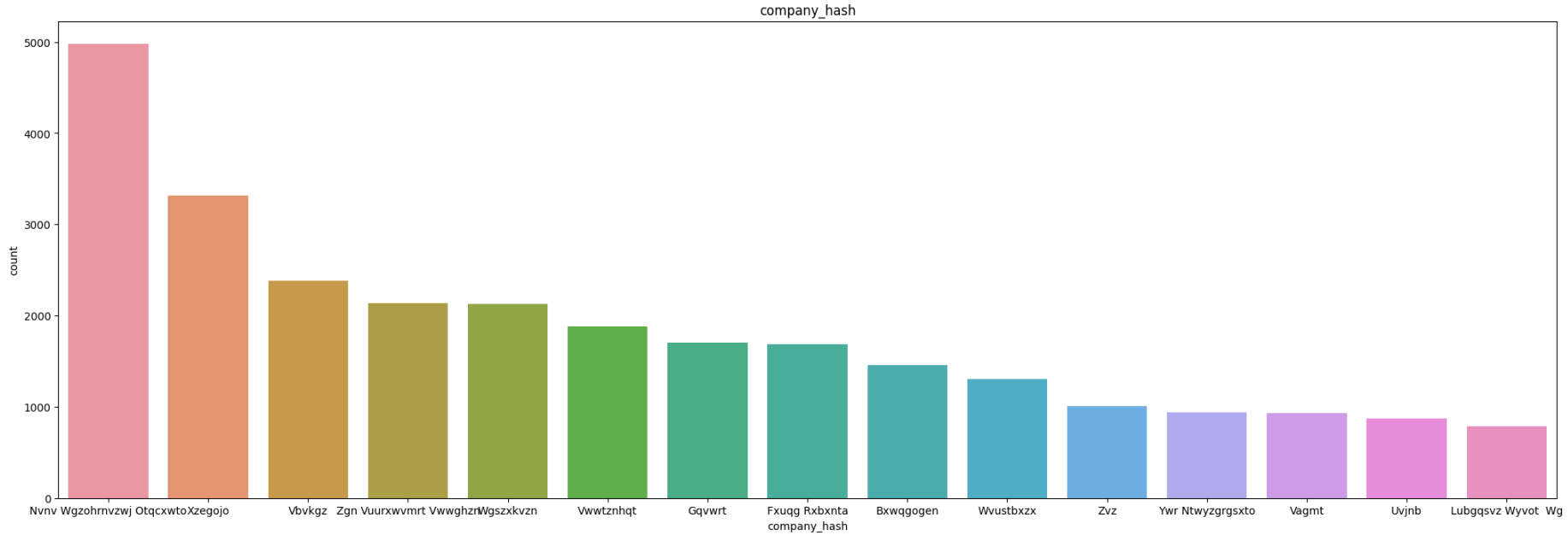
Univariate Analysis

Plotting Categorical Features

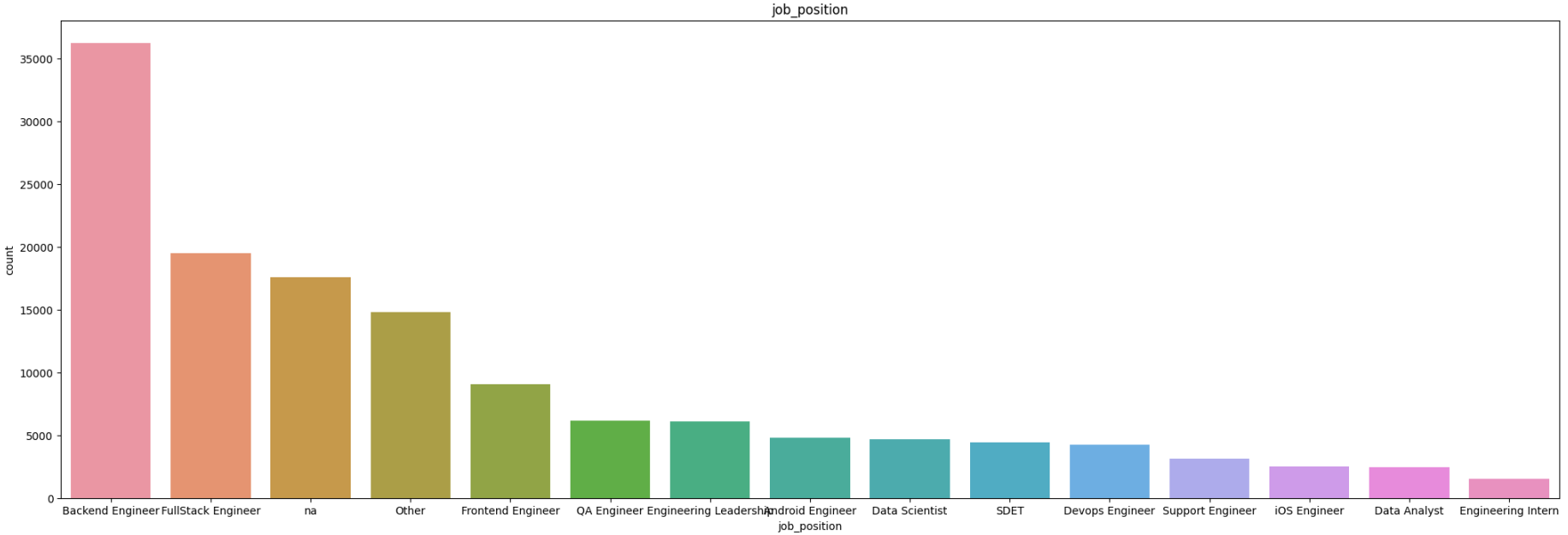
In [29]:
```
categroical_columns = [ 'company_hash','job_position','orgyear','ctc_updated_year']
```

In [30]:
```
for i in categroical_columns:
    tmp = data.copy()
    tmp['count'] = 1
    tmp = tmp.groupby(i).sum()['count'].reset_index().sort_values('count',ascending=False).head(15)
    plt.figure(figsize=(25,8))
    sns.barplot(data=tmp,y='count',x=i).set(title=i)

    plt.show()
```

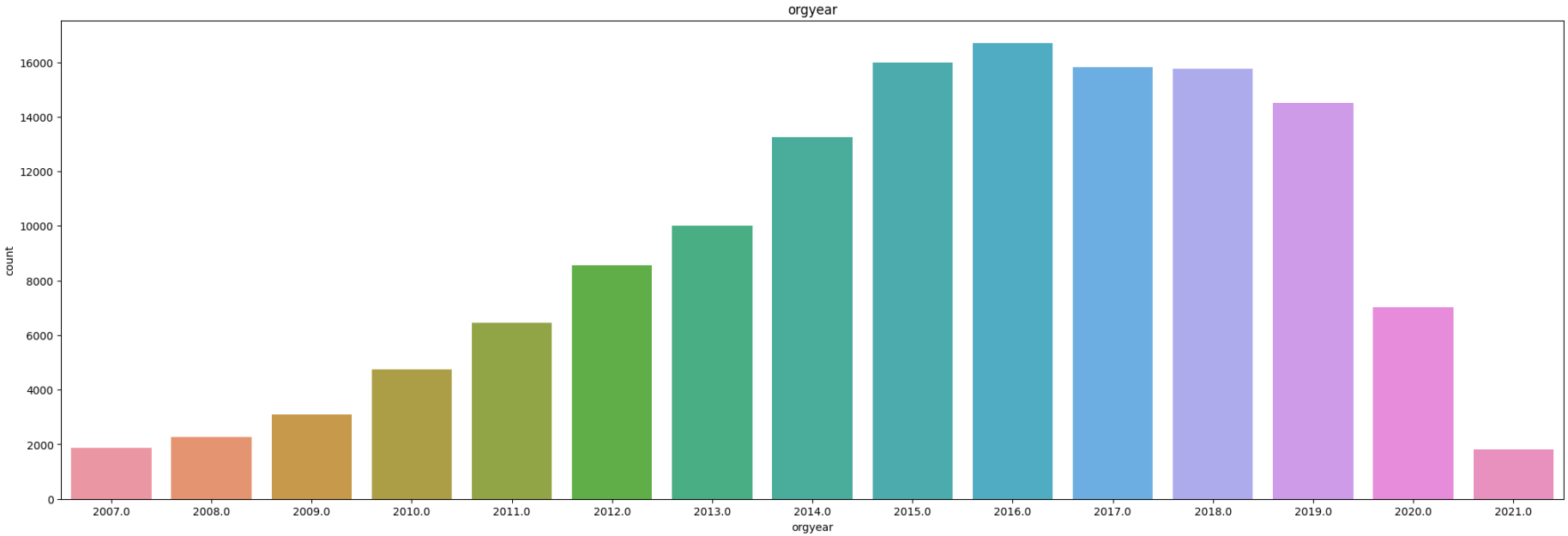
```
C:\Users\deepa\AppData\Local\Temp\ipykernel_17216\2639222101.py:4: FutureWarning: The default value of numeric_only in DataFram
eGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
  tmp = tmp.groupby(i).sum()['count'].reset_index().sort_values('count',ascending=False).head(15)
```



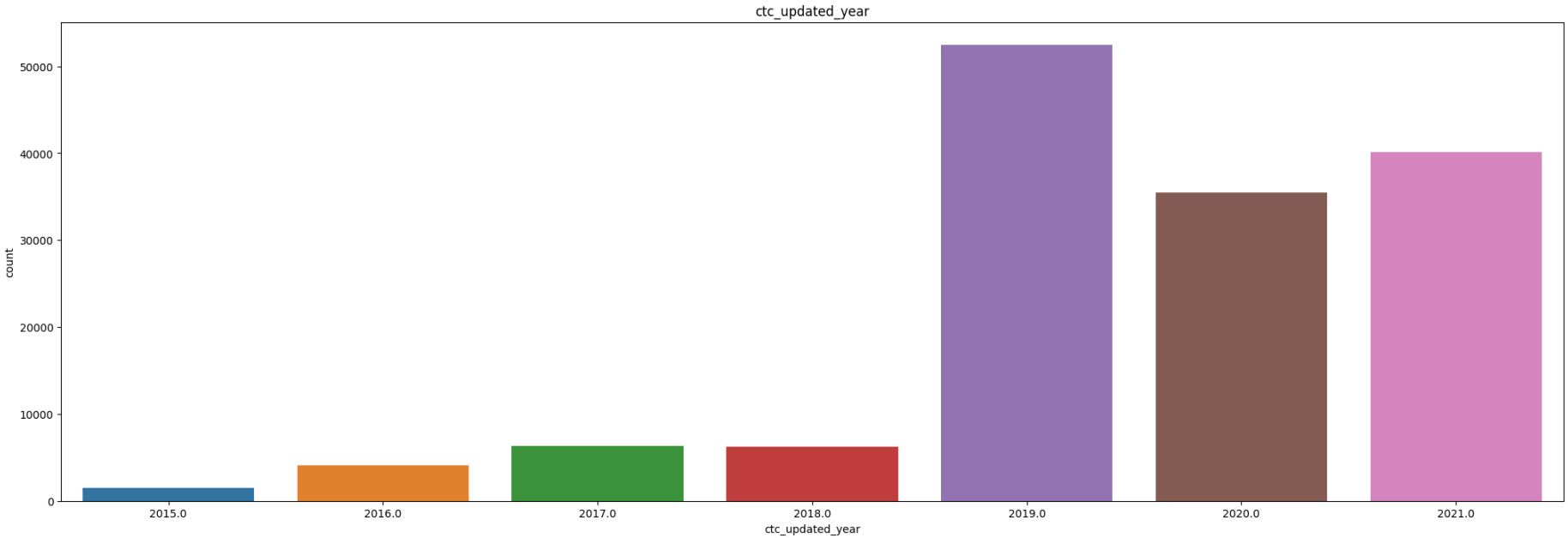
```
C:\Users\deepa\AppData\Local\Temp\ipykernel_17216\2639222101.py:4: FutureWarning: The default value of numeric_only in DataFram
eGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
  tmp = tmp.groupby(i).sum()['count'].reset_index().sort_values('count',ascending=False).head(15)
```




```
C:\Users\deepa\AppData\Local\Temp\ipykernel_17216\2639222101.py:4: FutureWarning: The default value of numeric_only in DataFram
eGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
  tmp = tmp.groupby(i).sum()['count'].reset_index().sort_values('count',ascending=False).head(15)
```

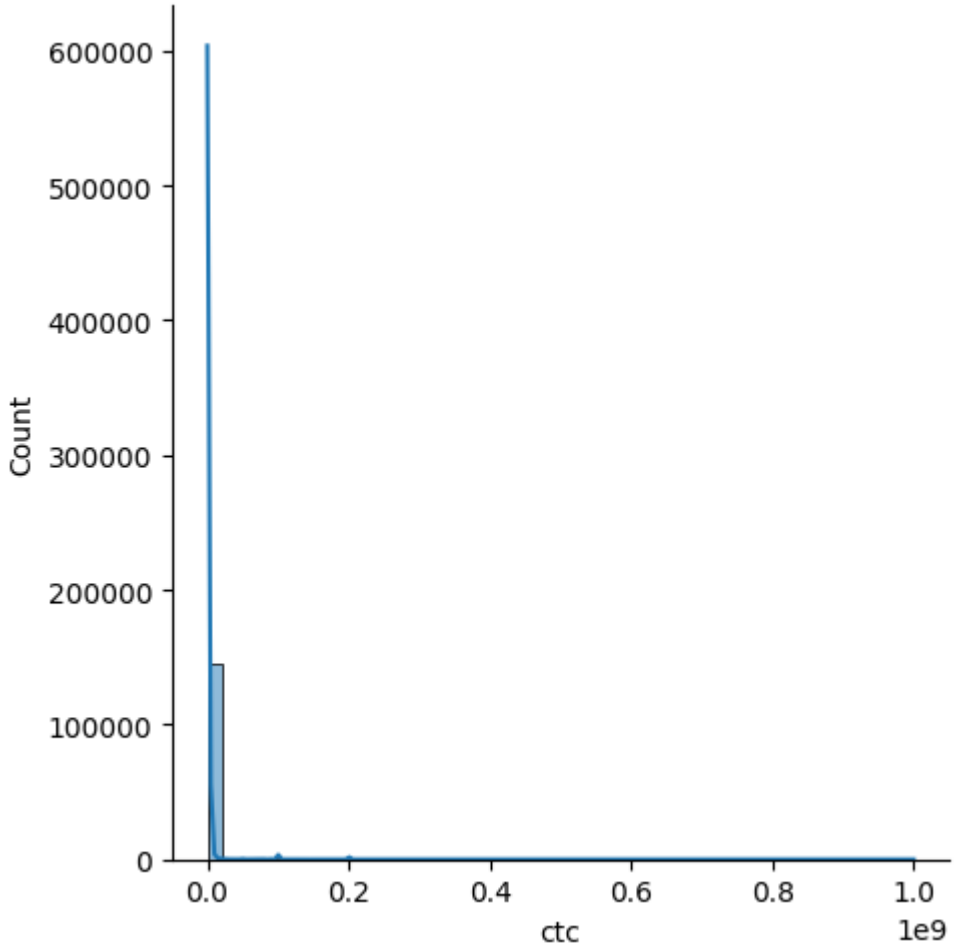



```
C:\Users\deepa\AppData\Local\Temp\ipykernel_17216\2639222101.py:4: FutureWarning: The default value of numeric_only in DataFram
eGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
  tmp = tmp.groupby(i).sum()['count'].reset_index().sort_values('count',ascending=False).head(15)
```



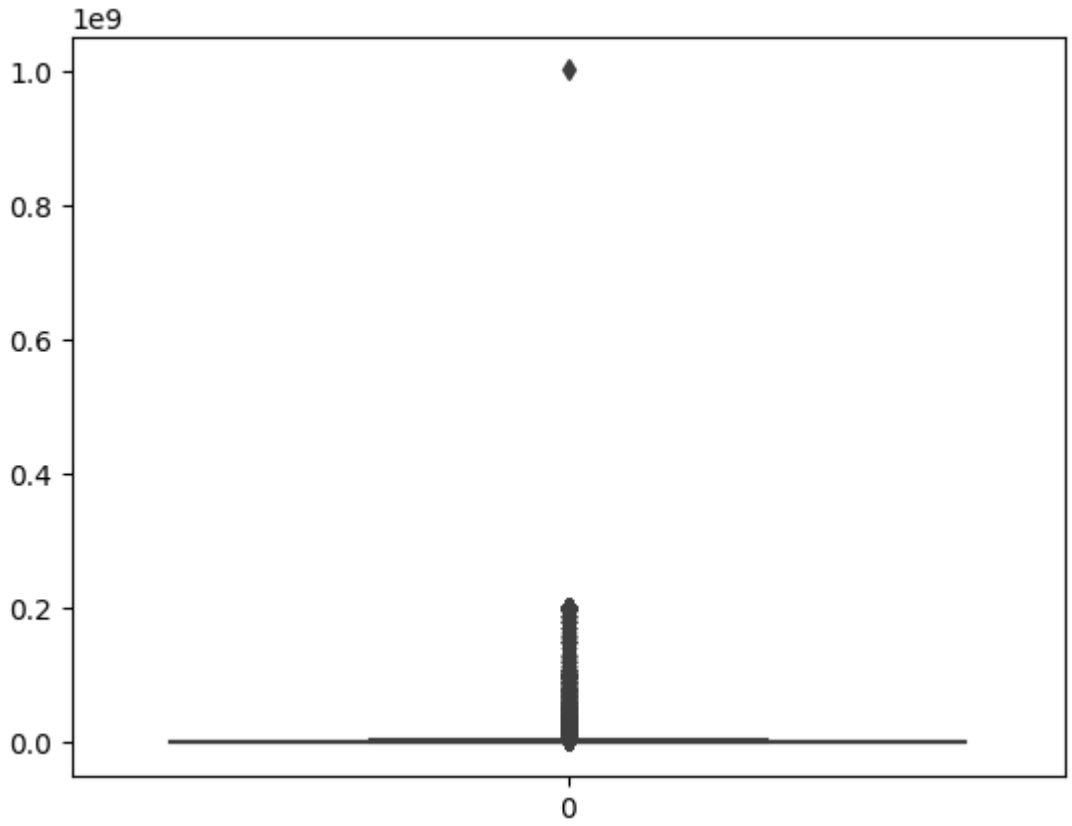
Plotting Continuous Features

In [31]:
```
sns.displot(data['ctc'],kde=True,bins=50)
plt.show()
```



The plot seems to be having large range of values, let's try to scale column for visualizing.

In [32]:
```
v = data['ctc']
#v = (v-v.mean())/v.std()
sns.boxplot(v)
plt.show()
```



In [33]:
```
data.sort_values(['ctc']).iloc[1000:1020,:]
```

Out[33]:

	email_hash	Unnamed: 0	company_hash	orgyear	ctc	job_position	ctc_updated_year
40428	439134c4b243fe33a240265f94d0c9d6e120f31cb6e2cc...	98842	Trxztdn	2019.0	20000	QA Engineer	2019.0
30954	3394eeca520d9029ce6bd56e83faa5d4d82c396f453e2a...	97003	Mrvmmtq	2011.0	20000	Android Engineer	2019.0
78889	83d1ece927b7d7e944454483a4a8a3b08a18ed846890ee...	88544	Oxbgz Eqvotq Hzxctqoxnj	2016.0	20000	Other	2017.0
90663	975e224e718de0d75c2d33d2bf24e75c4b7559664763b1...	125784	Wgcvrtzot Ntwyzgrgsxto	2016.0	20000	Backend Engineer	2019.0
74910	7d43b749f1651cba0a16743d6235f5f32a9a58837ee081...	24380	Ovu	2008.0	20000	QA Engineer	2019.0
38801	40bf699c6ab3273acfdda2c7922b30dbc5f8cedea903f4...	93058	Zgzt	2008.0	20000	FullStack Engineer	2019.0
25247	29e7d12a6225aeff7b5d92947f949e6646ec58111e5a5a...	4290	Vqttzv Bhrnxbtaxv	2001.0	20000	Other	2021.0
117820	c4a7229bb63eade33516411b68d8501420e15987d116eb...	135189	Uyvqbtvoj	2012.0	20000	Backend Engineer	2019.0
73669	7b3b93e56fd67d3ddfeb41f68a88a61d01c397aab9bbe1...	151028	Bvqxogen	2017.0	20000	na	2019.0
71300	774a4a75a15f53e19bbf4ef85a5c6dc3da689e4291f773...	77722	Rxetonjrt Xzntqzvnxgzvr Ucn Rna	2010.0	20000	Other	2016.0
120930	c9f0c1b5a2a71b0b754abcf68bc68c55188763ed0c8837...	99951	Wtrxsg Xzw	2009.0	20000	Backend Engineer	2019.0
5376	08bac5026bf379045813ce0e99e8df5601a56d913424fe...	84968	Uqtowqxmtq360 Ogrhnxgzo	2013.0	20000	FullStack Engineer	2019.0
49442	52cad1776d3a469cc3f056fb9568b0d6f8a2f901d4f857...	164185	Ougqnqvavq Vs	2008.0	20000	na	2016.0
36095	3c29ecc4a921bdf1f8ed0e3143069c47f836de0ad831fc...	100559	Xzntsqvnta Ntwyzgrgsj Sqghu	2016.0	20000	Android Engineer	2019.0

	email_hash	Unnamed: 0	company_hash	orgyear	ctc	job_position	ctc_updated_year
23539	270481314e2a3f52a67719a20b7b241090d1175067c527...	107733	Xburtdbt	2013.0	20000	Backend Engineer	2020.0
18584	1eb23fa16469ad4ec4bfd9c02a4878bee17986453eb87a...	81272	Gqvzst	2018.0	20000	FullStack Engineer	2019.0
139896	e9867fcd4ad217d5b3eb4be37e1770de27b9b1055e2a4c...	50112	Tuvb Ojontbo	2008.0	20000	Backend Engineer	2016.0
35678	3b71a41715519c7a82a0eee25b9e90edd749d77852fe55...	46366	Ogenwvqt	2004.0	20000	Backend Engineer	2017.0
131924	dc43256af03d641e65dc34d57ea0872500935062f7f20a...	142496	Vbvkgz	2014.0	20000	Other	2017.0
107673	b3a96ff0081272c4627cdaa0c85b4b4f24b5c46792b035...	166905	Mhoxztoowgat Sbmv	2015.0	20000	Data Scientist	2019.0

In [34]:
```
data = data[data['ctc'] >702475]
```

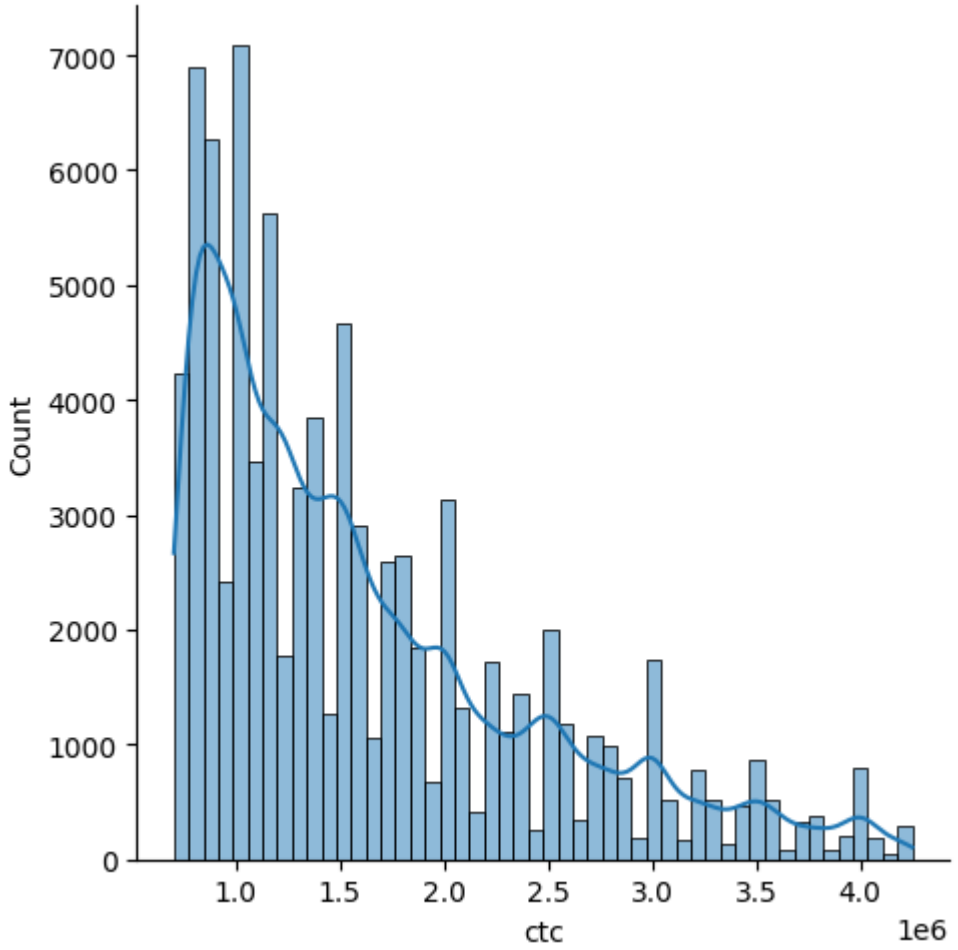
Outlier Removal using IQR

In [35]:
```
dftmp = data.copy()
print(dftmp.shape)
cols = ['ctc'] # one or more

Q1 = dftmp[cols].quantile(0.25)
Q3 = dftmp[cols].quantile(0.75)
IQR = Q3 - Q1

dftmp = dftmp[~((dftmp[cols] < (Q1 - 1.5 * IQR)) |(dftmp[cols] > (Q3 + 1.5 * IQR))).any(axis=1)]
print(dftmp.shape)
```

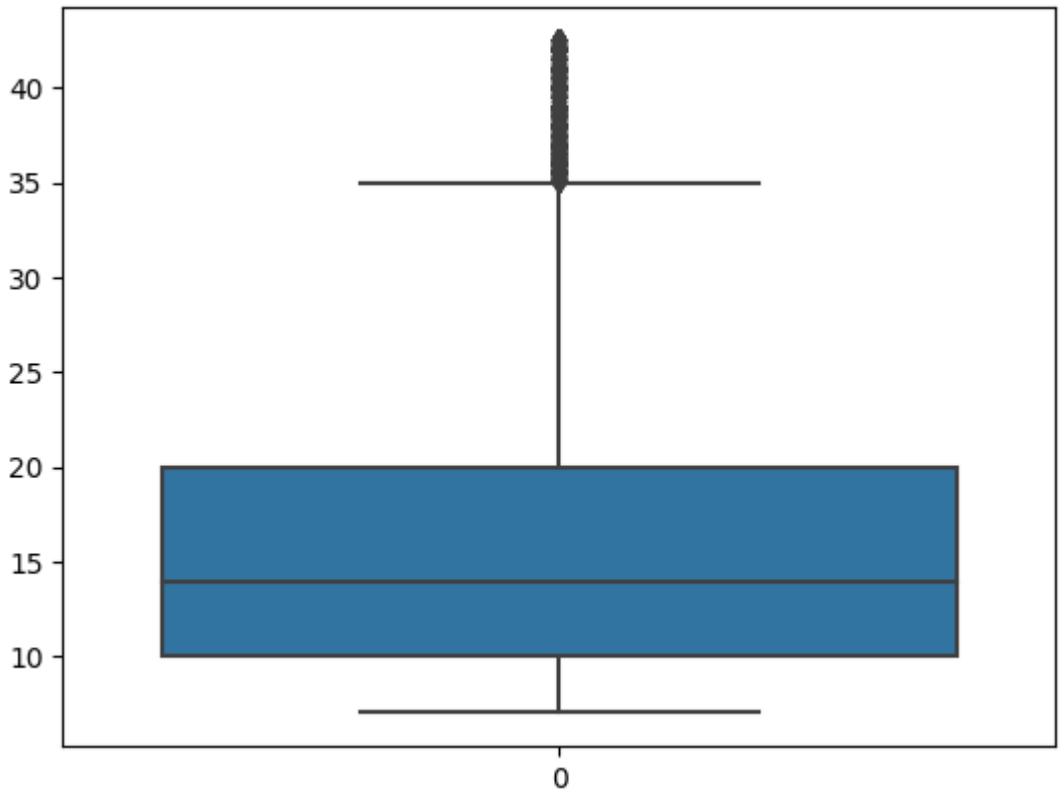
```
(92558, 14)
(86464, 14)
```

In [36]:
```
#dftmp = dftmp[dftmp['ctc']>300000]
```

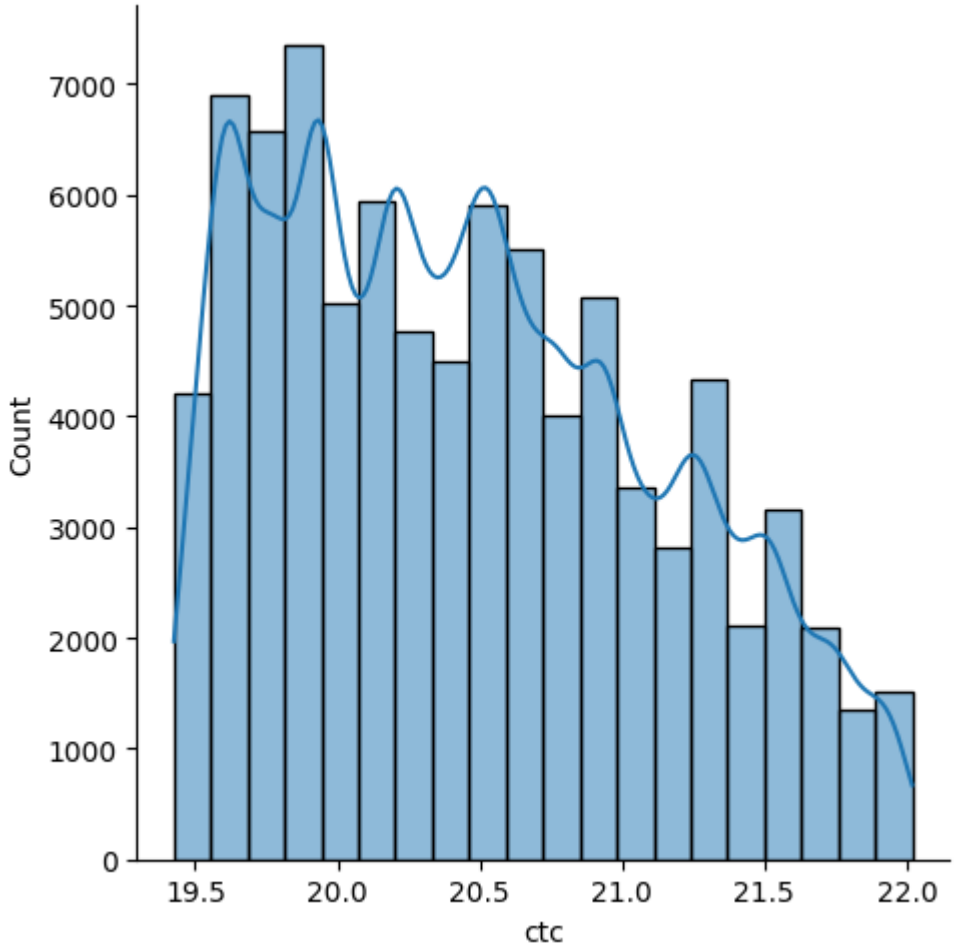
In [37]:
```
v = dftmp['ctc']
sns.displot(v,kde=True,bins=50)
plt.show()
```



In [38]:
```
v = dftmp['ctc']/100000
#v = (v-v.mean())/v.std()
sns.boxplot(v)
plt.show()
```



In [39]:
```
v = np.log2(dftmp['ctc'])
sns.displot(v,kde=True,bins=20)
plt.show()
```



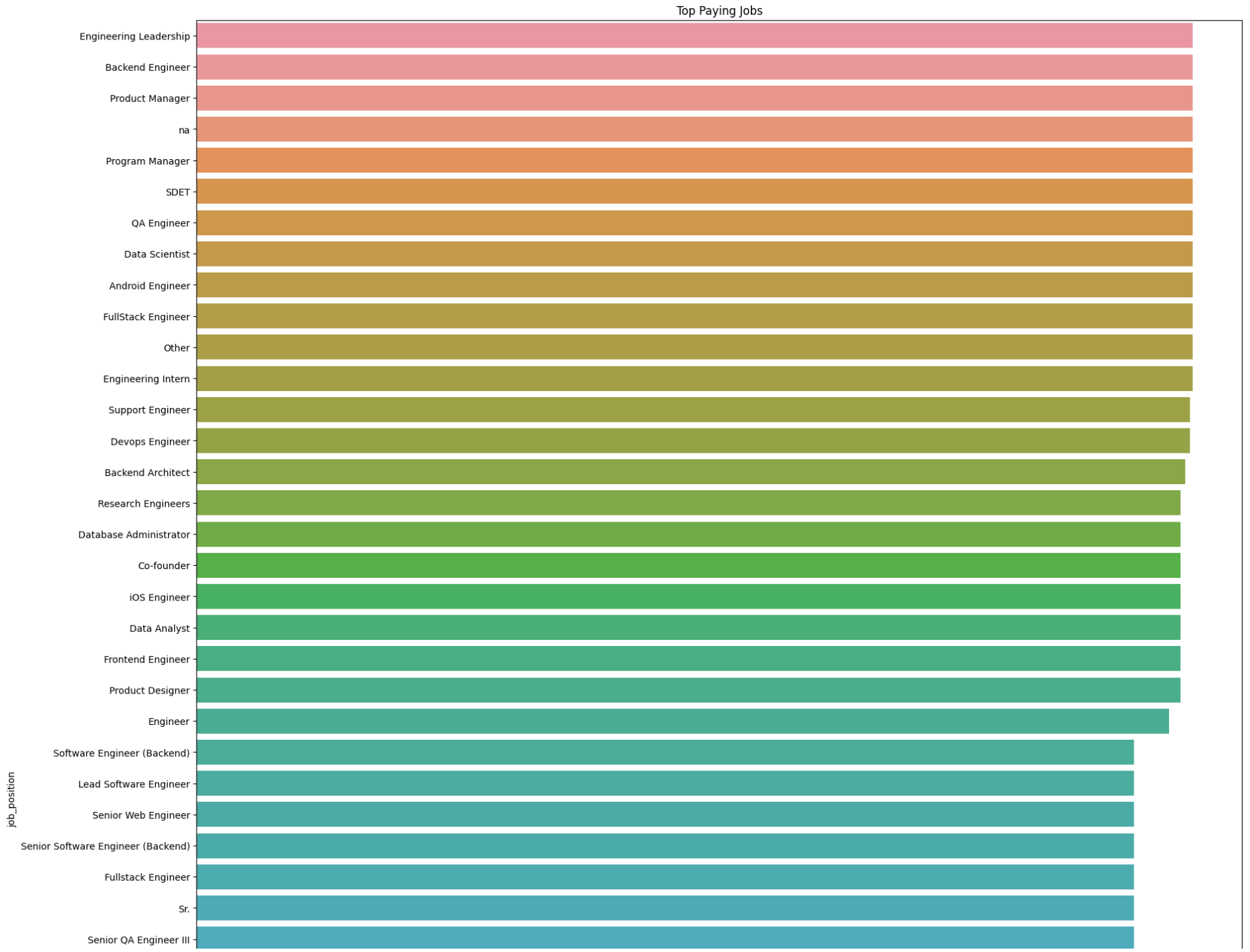
In [40]:
```
dateda = dftmp.copy()
```

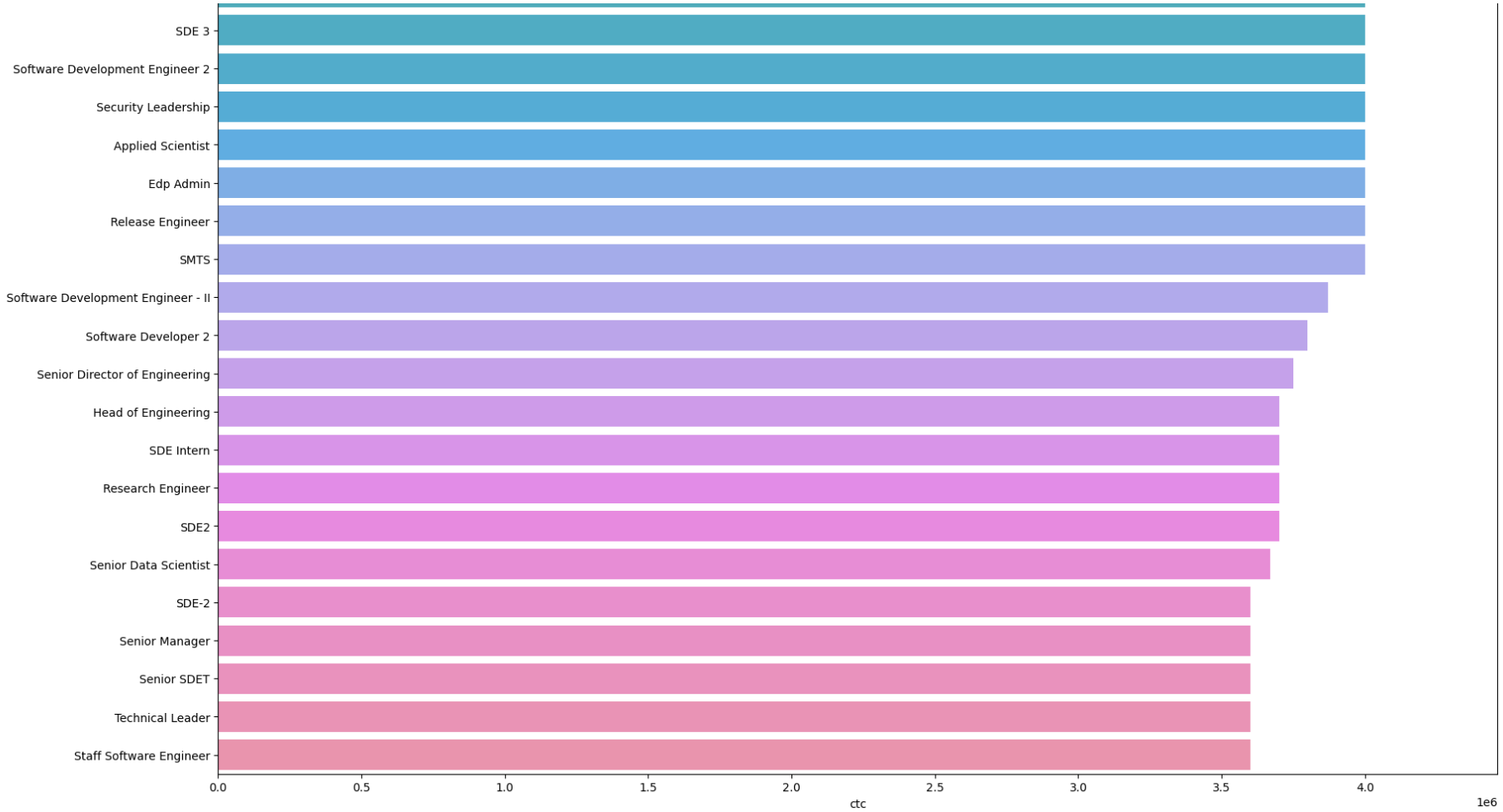
In [ ]:
```

```

Multivariate Analysis

In [43]:
```
tmp = dftmp.copy()
tmp = tmp.groupby(['job_position']).max()['ctc'].reset_index().sort_values('ctc',ascending=False).head(50)
```

```
plt.figure(figsize=(20,30))
sns.barplot(data=tmp,x='ctc',y='job_position').set(title="Top Paying Jobs")
plt.show()
list(tmp['job_position'])
```


Top Paying Jobs
job_position
Engineering Leadership
Backend Engineer
Product Manager
na
Program Manager
SDET
QA Engineer
Data Scientist
Android Engineer
FullStack Engineer
Other
Engineering Intern
Support Engineer
Devops Engineer
Backend Architect
Research Engineers
Database Administrator
Co-founder
iOS Engineer
Data Analyst
Frontend Engineer
Product Designer
Engineer
Software Engineer (Backend)
Lead Software Engineer
Senior Web Engineer
Senior Software Engineer (Backend)
Fullstack Engineer
Sr.
Senior QA Engineer III

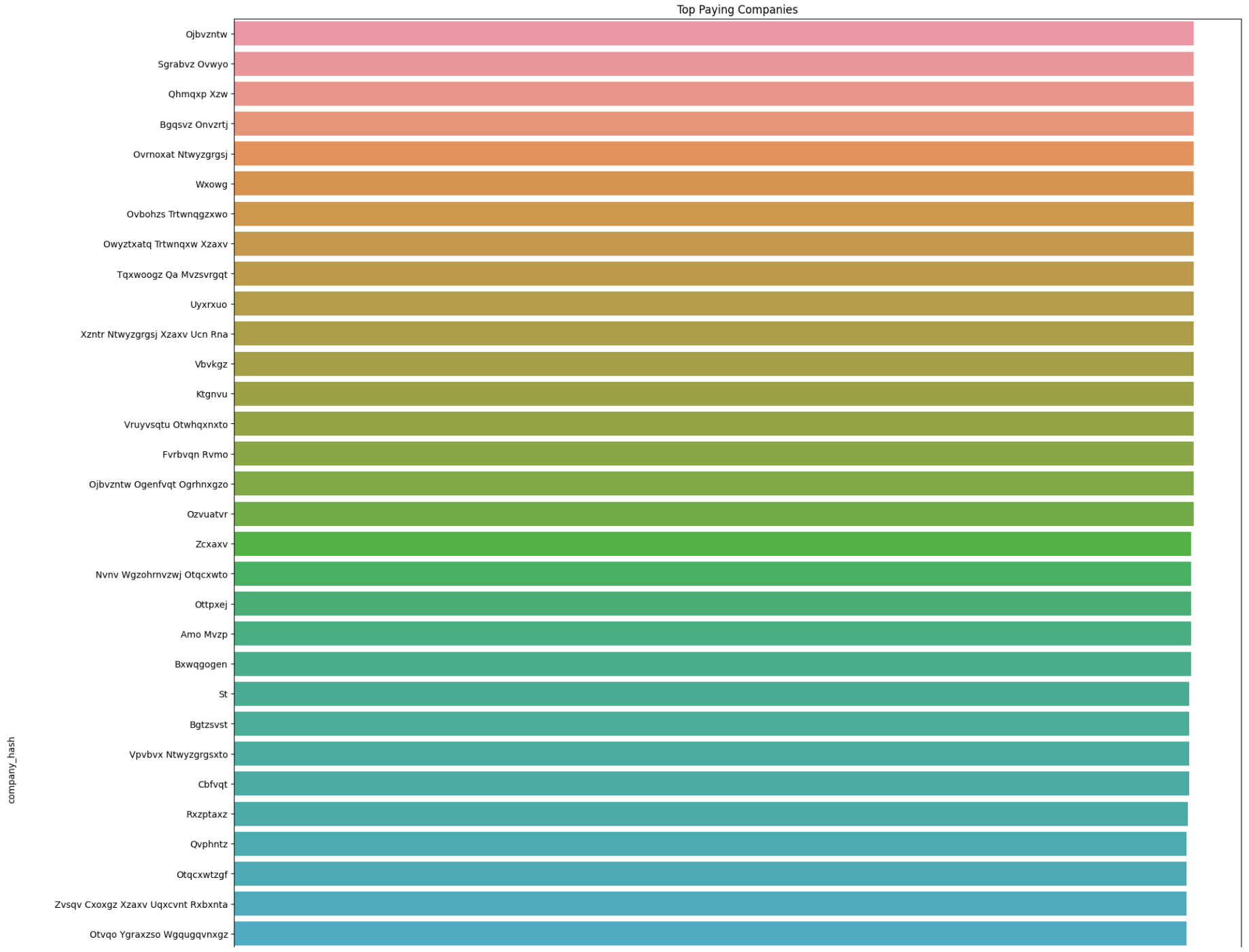

SDE 3
Software Development Engineer 2
Security Leadership
Applied Scientist
Edp Admin
Release Engineer
SMTS
Software Development Engineer - II
Software Developer 2
Senior Director of Engineering
Head of Engineering
SDE Intern
Research Engineer
SDE2
Senior Data Scientist
SDE-2
Senior Manager
Senior SDET
Technical Leader
Staff Software Engineer
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
1e6
ctc

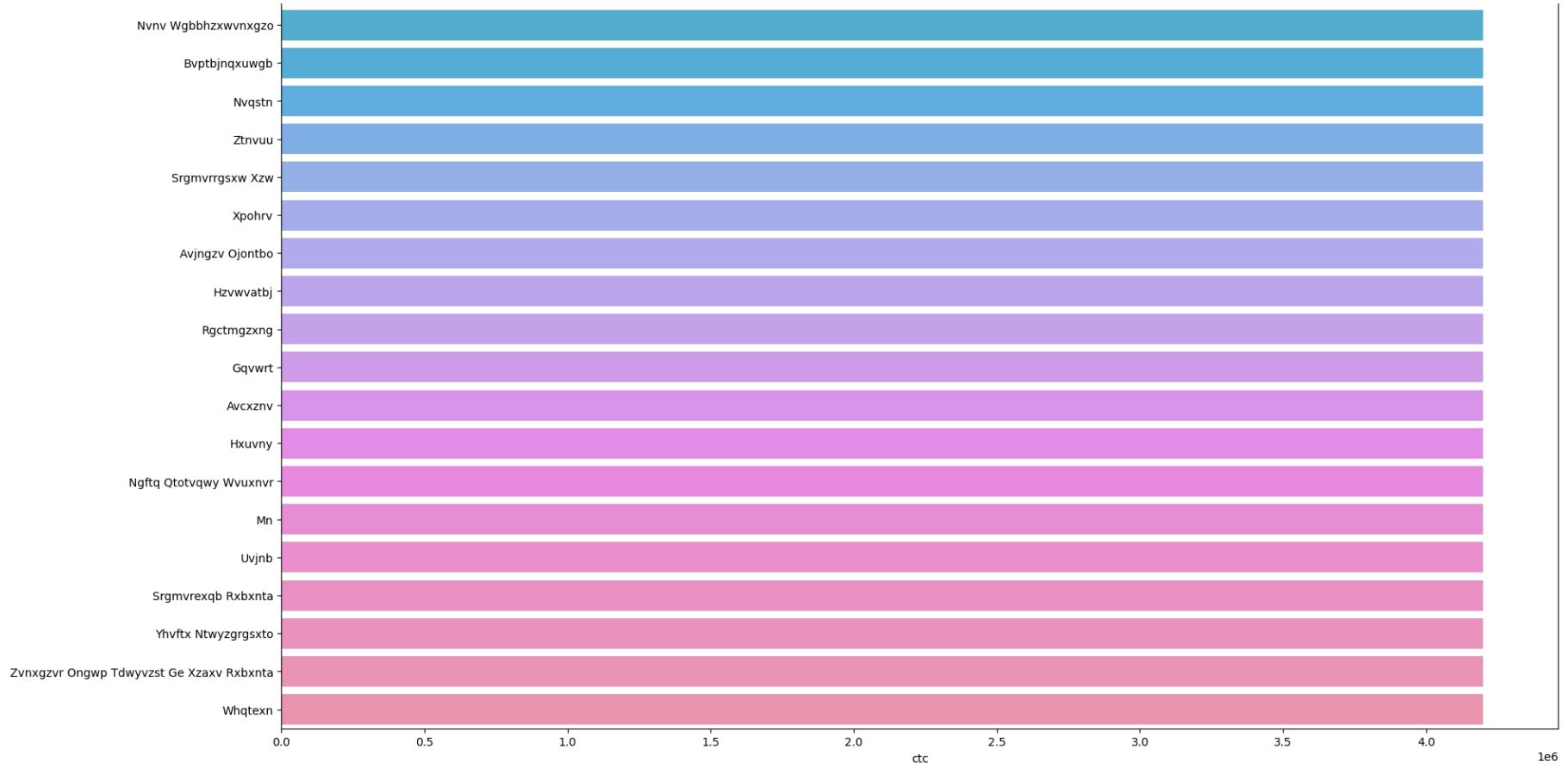
Out[43]:
```
['Engineering Leadership',
 'Backend Engineer',
 'Product Manager',
 'na',
 'Program Manager',
 'SDET',
 'QA Engineer',
 'Data Scientist',
 'Android Engineer',
 'FullStack Engineer',
 'Other',
 'Engineering Intern',
 'Support Engineer',
 'Devops Engineer',
 'Backend Architect',
 'Research Engineers',
 'Database Administrator',
 'Co-founder',
 'iOS Engineer',
 'Data Analyst',
 'Frontend Engineer',
 'Product Designer',
 'Engineer',
 'Software Engineer (Backend)',
 'Lead Software Engineer',
 'Senior Web Engineer',
 'Senior Software Engineer (Backend)',
 'Fullstack Engineer',
 'Sr.',
 'Senior QA Engineer III',
 'SDE 3',
 'Software Development Engineer 2',
 'Security Leadership',
 'Applied Scientist',
 'Edp Admin',
 'Release Engineer',
 'SMTS',
 'Software Development Engineer - II',
 'Software Developer 2',
 'Senior Director of Engineering',
```

```
 'Head of Engineering',
 'SDE Intern',
 'Research Engineer',
 'SDE2',
 'Senior Data Scientist',
 'SDE-2',
 'Senior Manager',
 'Senior SDET',
 'Technical Leader',
 'Staff Software Engineer']
```

In [80]:

```
tmp = dftmp.copy()
tmp = tmp.groupby(['company_hash']).max()['ctc'].reset_index().sort_values('ctc',ascending=False).head(50)
plt.figure(figsize=(20,30))
sns.barplot(data=tmp,x='ctc',y='company_hash').set(title="Top Paying Companies")
plt.show()

list(tmp['company_hash'])
```


Top Paying Companies
company_hash
Ojbvzntw
Sgrabvz Ovwyo
Qhmqxp Xzw
Bgqsvz Onvzrtj
Ovrnoxat Ntwyzgrgsj
Wxowg
Ovbohzs Trtwnqgzxwo
Owyztxatq Trtwnqxw Xzaxv
Tqxwoogz Qa Mvzsvrgqt
Uyxrxuo
Xzntr Ntwyzgrgsj Xzaxv Ucn Rna
Vbvkgz
Ktgnvu
Vruyvsqtu Otwhqxnxto
Fvrbvqn Rvmo
Ojbvzntw Ogenfvqt Ogrhnxgzo
Ozvuatvr
Zcxaxv
Nvnv Wgzohrnvzwj Otqcxwto
Ottpxej
Amo Mvzp
Bxwqgogen
St
Bgtzsvst
Vpvbvx Ntwyzgrgsxto
Cbfvqt
Rxzptaxz
Qvphntz
Otqcxwtzgf
Zvsqv Cxoxgz Xzaxv Uqxcvnt Rxbxnta
Otvqo Ygraxzso Wgqugqvnxgz


Nvnv Wgbbhzxwvnxgzo
Bvptbjnqxuwgb
Nvqstn
Ztnvuu
Srgmvrrgsxw Xzw
Xpohrv
Avjngzv Ojontbo
Hzvwvatbj
Rgctmgzxng
Gqvwrt
Avcxznv
Hxuvny
Ngftq Qtotvqwy Wvuxnvr
Mn
Uvjnb
Srgmvrexqb Rxbxnta
Yhvftx Ntwyzgrgsxto
Zvnxgzvr Ongwp Tdwyvzst Ge Xzaxv Rxbxnta
Whqtexn
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
1e6
ctc

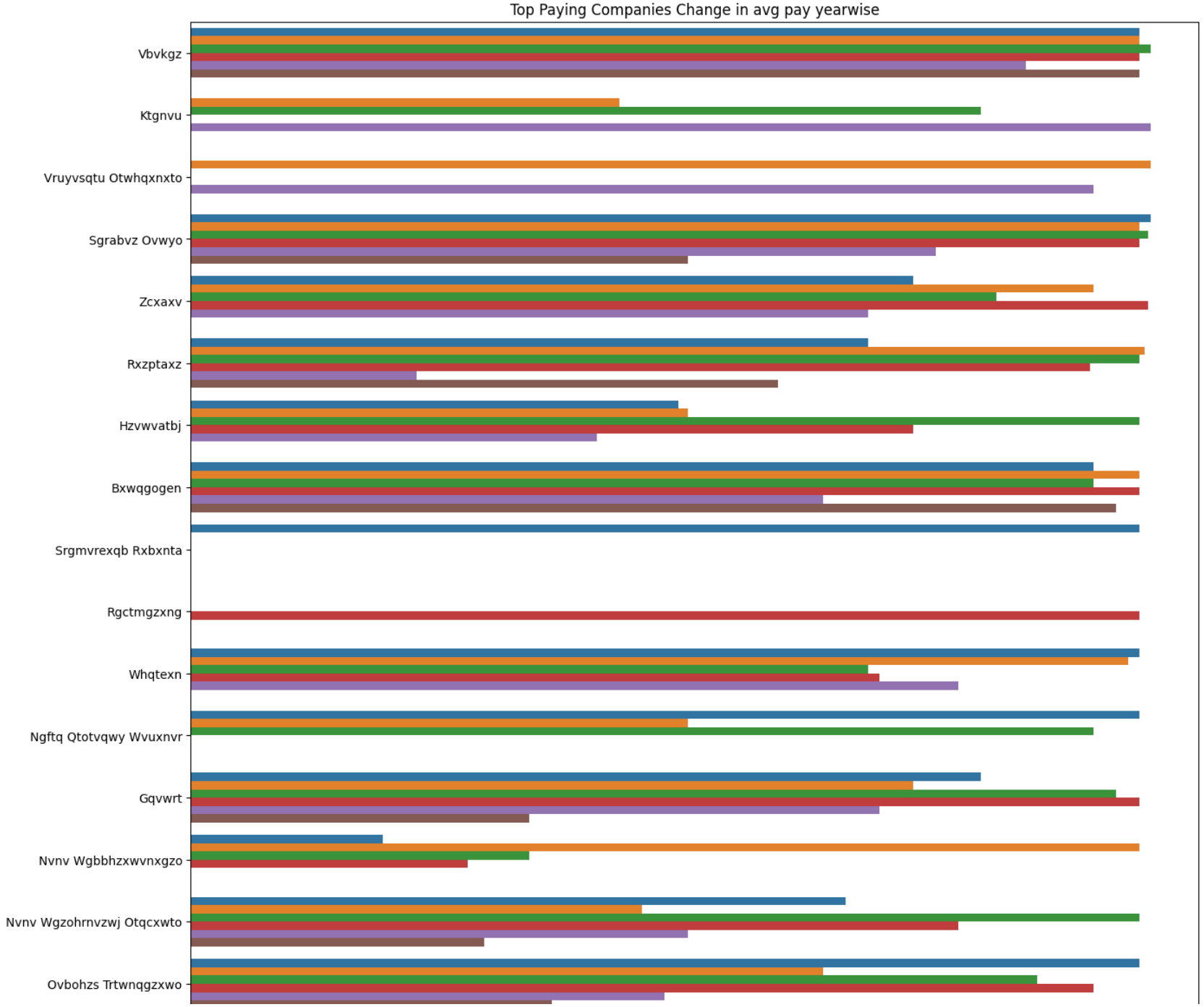
Out[80]:
```
['Ojbvzntw',
 'Sgrabvz Ovwyo',
 'Qhmqxp Xzw',
 'Bgqsvz Onvzrtj',
 'Ovrnoxat Ntwyzgrgsj',
 'Wxowg',
 'Ovbohzs Trtwnqgzxwo',
 'Owyztxatq Trtwnqxw Xzaxv',
 'Tqxwoogz Qa Mvzsvrgqt',
 'Uyxrxuo',
 'Xzntr Ntwyzgrgsj Xzaxv Ucn Rna',
 'Vbvkgz',
 'Ktgnvu',
 'Vruyvsqtu Otwhqxnxto',
 'Fvrbvqn Rvmo',
 'Ojbvzntw Ogenfvqt Ogrhnxgzo',
 'Ozvuatvr',
 'Zcxaxv',
 'Nvnv Wgzohrnvzwj Otqcxwto',
 'Ottpxej',
 'Amo Mvzp',
 'Bxwqgogen',
 'St',
 'Bgtzsvst',
 'Vpvbvx Ntwyzgrgsxto',
 'Cbfvqt',
 'Rxzptaxz',
 'Qvphntz',
 'Otqcxwtzgf',
 'Zvsqv Cxoxgz Xzaxv Uqxcvnt Rxbxnta',
 'Otvqo Ygraxzso Wgqugqvnxgz',
 'Nvnv Wgbbhzxwvnxgzo',
 'Bvptbjnqxuwgb',
 'Nvqstn',
 'Ztnvuu',
 'Srgmvrrgsxw Xzw',
 'Xpohrv',
 'Avjngzv Ojontbo',
 'Hzvwvatbj',
 'Rgctmgzxng',
```

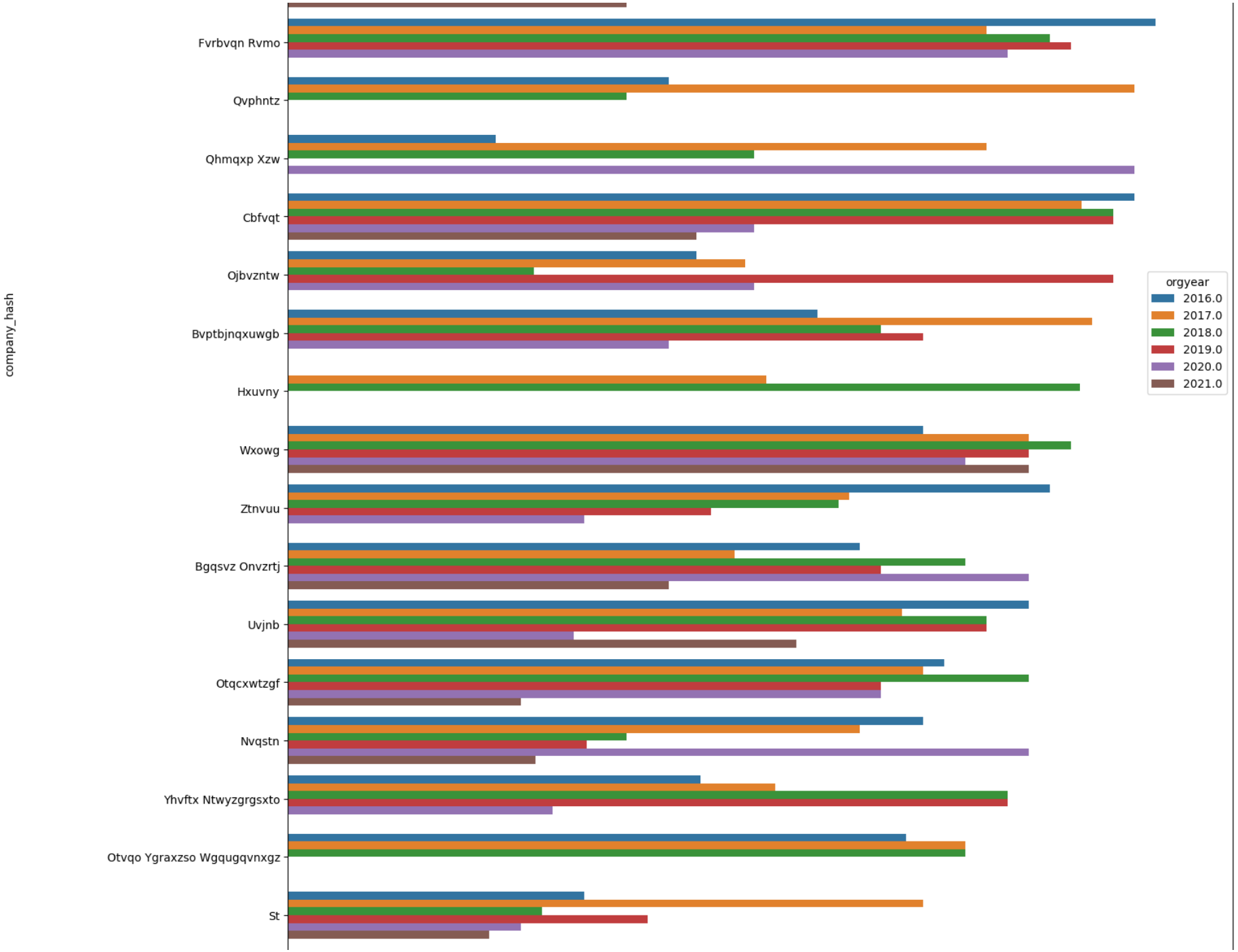
```
 'Gqvwrt',
 'Avcxznv',
 'Hxuvny',
 'Ngftq Qtotvqwy Wvuxnvr',
 'Mn',
 'Uvjnb',
 'Srgmvrexqb Rxbxnta',
 'Yhvftx Ntwyzgrgsxto',
 'Zvnxgzvr Ongwp Tdwyvzst Ge Xzaxv Rxbxnta',
 'Whqtexn']
```

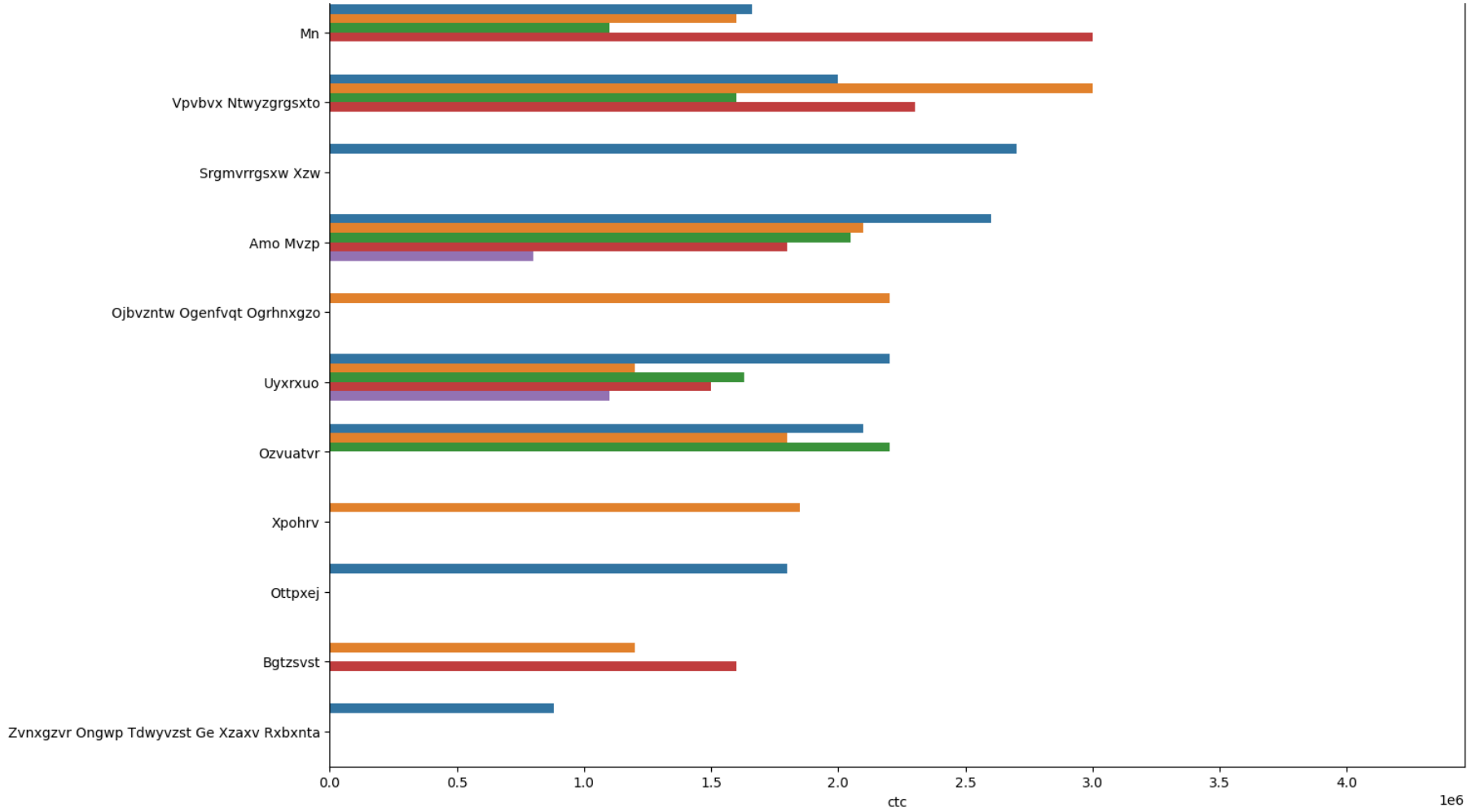
In [45]:

```
tmp = dftmp.copy()
tmp = tmp[tmp['company_hash'].isin(['Ojbvzntw',
 'Sgrabvz Ovwyo',
 'Qhmqxp Xzw',
 'Bgqsvz Onvzrtj',
 'Ovrnoxat Ntwyzgrgsj',
 'Wxowg',
 'Ovbohzs Trtwnqgzxwo',
 'Owyztxatq Trtwnqxw Xzaxv',
 'Tqxwoogz Qa Mvzsvrgqt',
 'Uyxrxuo',
 'Xzntr Ntwyzgrgsj Xzaxv Ucn Rna',
 'Vbvkgz',
 'Ktgnvu',
 'Vruyvsqtu Otwhqxnxto',
 'Fvrbvqn Rvmo',
 'Ojbvzntw Ogenfvqt Ogrhnxgzo',
 'Ozvuatvr',
 'Zcxaxv',
 'Nvnv Wgzohrnvzwj Otqcxwto',
 'Ottpxej',
 'Amo Mvzp',
 'Bxwqgogen',
 'St',
 'Bgtzsvst',
 'Vpvbvx Ntwyzgrgsxto',
 'Cbfvqt',
 'Rxzptaxz',
 'Qvphntz',
 'Otqcxwtzgf',
```

```
 'Zvsqv Cxoxgz Xzaxv Uqxcvnt Rxbxnta',
 'Otvqo Ygraxzso Wgqugqvnxgz',
 'Nvnv Wgbbhzxwvnxgzo',
 'Bvptbjnqxuwgb',
 'Nvqstn',
 'Ztnvuu',
 'Srgmvrrgsxw Xzw',
 'Xpohrv',
 'Avjngzv Ojontbo',
 'Hzvwvatbj',
 'Rgctmgzxng',
 'Gqvwrt',
 'Avcxznv',
 'Hxuvny',
 'Ngftq Qtotvqwy Wvuxnvr',
 'Mn',
 'Uvjnb',
 'Srgmvrexqb Rxbxnta',
 'Yhvftx Ntwyzgrgsxto',
 'Zvnxgzvr Ongwp Tdwyvzst Ge Xzaxv Rxbxnta',
 'Whqtexn'])]
tmp = tmp[tmp['orgyear'] >= 2016]
tmp = tmp.groupby(['company_hash','orgyear']).max()['ctc'].reset_index().sort_values('ctc',ascending=False)
plt.figure(figsize=(15,40))
sns.barplot(data=tmp,x='ctc',y='company_hash',hue='orgyear').set(title="Top Paying Companies Change in avg pay yearwise")
plt.show()
```


Top Paying Companies Change in avg pay yearwise
Vbvkgz
Ktgnvu
Vruyvsqtu Otwhqxnxto
Sgrabvz Ovwyo
Zcxaxv
Rxzptaxz
Hzvwvatbj
Bxwqgogen
Srgmvrexqb Rxbxnta
Rgctmgzxng
Whqtexn
Ngftq Qtotvqwy Wvuxnvr
Gqvwrt
Nvnv Wgbbhzxwvnxgzo
Nvnv Wgzohrnvzwj Otqcxwto
Ovbohzs Trtwnqgzxwo

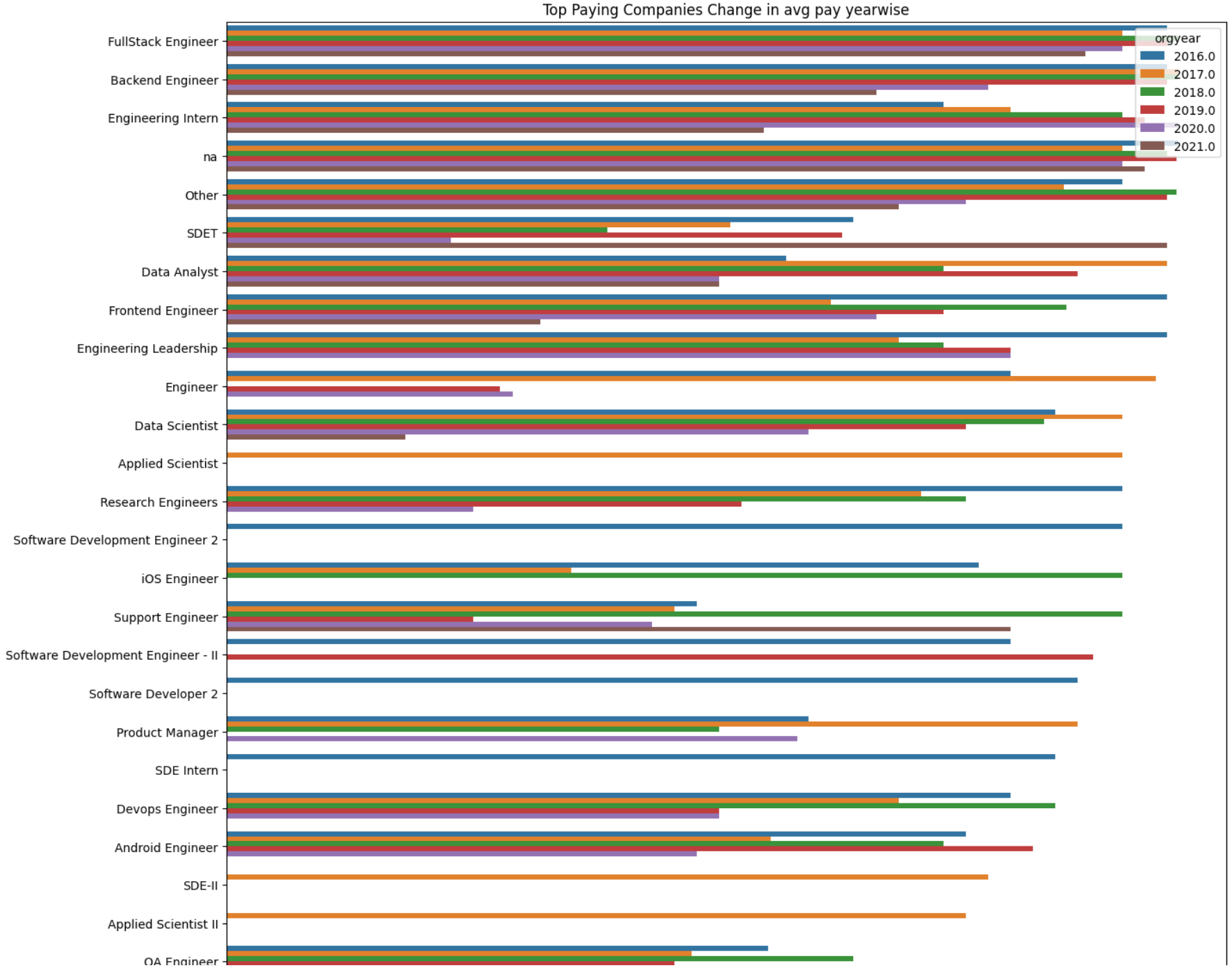

company_hash
Fvrbvqn Rvmo
Qvphntz
Qhmqxp Xzw
Cbfvqt
Ojbvzntw
Bvptbjnqxuwgb
Hxuvny
Wxowg
Ztnvuu
Bgqsvz Onvzrtj
Uvjnb
Otqcxwtzgf
Nvqstn
Yhvftx Ntwyzgrgsxto
Otvqo Ygraxzso Wgqugqvnxgz
St
orgyear
2016.0
2017.0
2018.0
2019.0
2020.0
2021.0

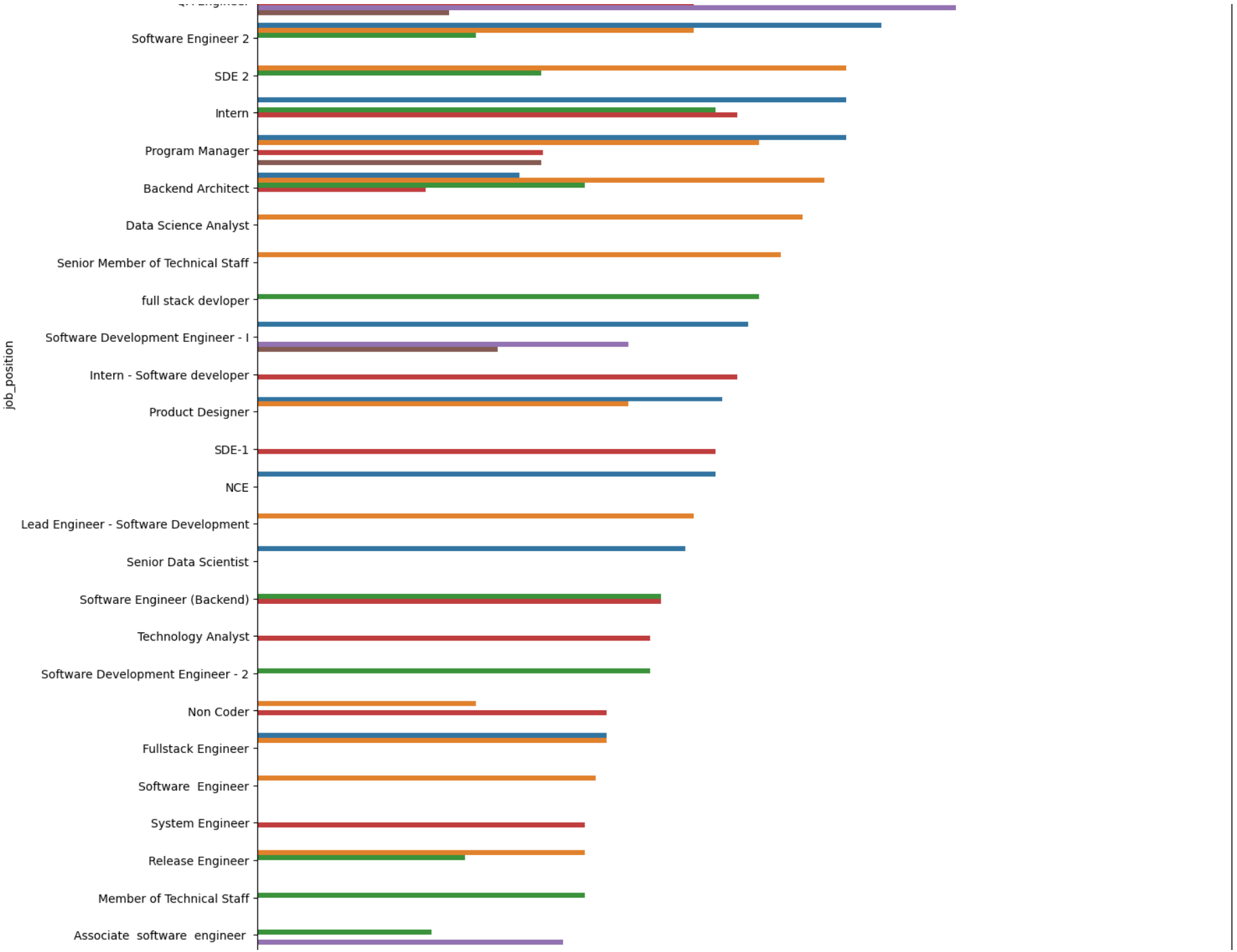


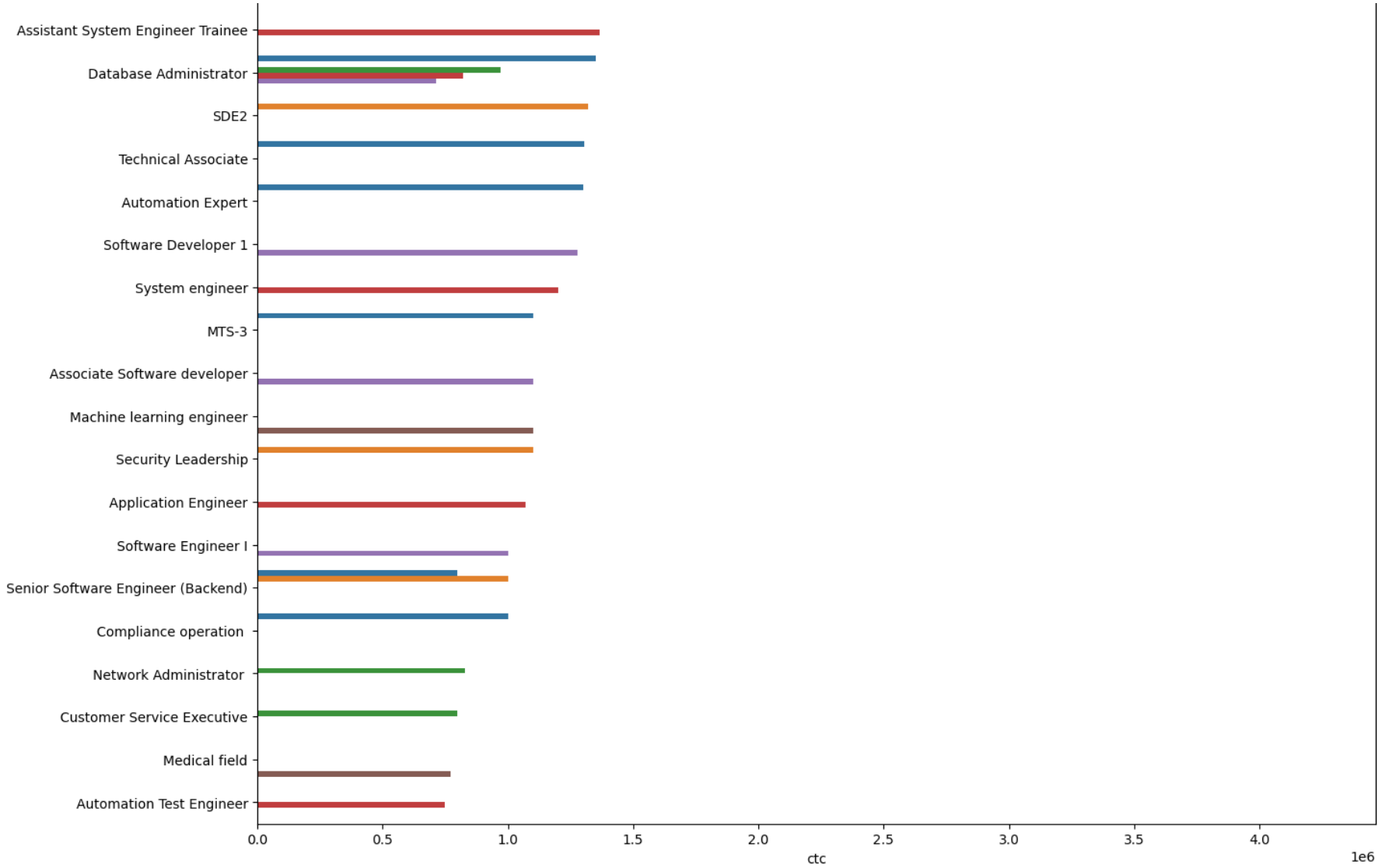
In [46]:
```
tmp = dftmp.copy()
tmp = tmp[tmp['company_hash'].isin(['Ojbvzntw',
 'Sgrabvz Ovwyo',
 'Qhmqxp Xzw',
 'Bgqsvz Onvzrtj',
 'Ovrnoxat Ntwyzgrgsj',
 'Wxowg',
 'Ovbohzs Trtwnqgzxwo',
 'Owyztxatq Trtwnqxw Xzaxv',
 'Tqxwoogz Qa Mvzsvrgqt',
```

```
'Uyxrxuo',
'Xzntr Ntwyzgrgsj Xzaxv Ucn Rna',
'Vbvkgz',
'Ktgnvu',
'Vruyvsqtu Otwhqxnxto',
'Fvrbvqn Rvmo',
'Ojbvzntw Ogenfvqt Ogrhnxgzo',
'Ozvuatvr',
'Zcxaxv',
'Nvnv Wgzohrnvzwj Otqcxwto',
'Ottpxej',
'Amo Mvzp',
'Bxwqgogen',
'St',
'Bgtzsvst',
'Vpvbvx Ntwyzgrgsxto',
'Cbfvqt',
'Rxzptaxz',
'Qvphntz',
'Otqcxwtzgf',
'Zvsqv Cxoxgz Xzaxv Uqxcvnt Rxbxnta',
'Otvqo Ygraxzso Wgqugqvnxgz',
'Nvnv Wgbbhzxwvnxgzo',
'Bvptbjnqxuwgb',
'Nvqstn',
'Ztnvuu',
'Srgmvrrgsxw Xzw',
'Xpohrv',
'Avjngzv Ojontbo',
'Hzvwvatbj',
'Rgctmgzxng',
'Gqvwrt',
'Avcxznv',
'Hxuvny',
'Ngftq Qtotvqwy Wvuxnvr',
'Mn',
'Uvjnb',
'Srgmvrexqb Rxbxnta',
'Yhvftx Ntwyzgrgsxto',
'Zvnxgzvr Ongwp Tdwyvzst Ge Xzaxv Rxbxnta',
'Whqtexn'])]
```

```
tmp = tmp[tmp['orgyear'] >= 2016]
tmp = tmp.groupby(['job_position','orgyear']).max()['ctc'].reset_index().sort_values('ctc',ascending=False)
plt.figure(figsize=(15,40))
sns.barplot(data=tmp,x='ctc',y='job_position',hue='orgyear').set(title="Top Paying Companies Change in avg pay yearwise")
plt.show()
```


Top Paying Companies Change in avg pay yearwise
orgyear
2016.0
2017.0
2018.0
2019.0
2020.0
2021.0
FullStack Engineer
Backend Engineer
Engineering Intern
na
Other
SDET
Data Analyst
Frontend Engineer
Engineering Leadership
Engineer
Data Scientist
Applied Scientist
Research Engineers
Software Development Engineer 2
iOS Engineer
Support Engineer
Software Development Engineer - II
Software Developer 2
Product Manager
SDE Intern
Devops Engineer
Android Engineer
SDE-II
Applied Scientist II
QA Engineer


job_position
Software Engineer 2
SDE 2
Intern
Program Manager
Backend Architect
Data Science Analyst
Senior Member of Technical Staff
full stack devloper
Software Development Engineer - I
Intern - Software developer
Product Designer
SDE-1
NCE
Lead Engineer - Software Development
Senior Data Scientist
Software Engineer (Backend)
Technology Analyst
Software Development Engineer - 2
Non Coder
Fullstack Engineer
Software Engineer
System Engineer
Release Engineer
Member of Technical Staff
Associate software engineer

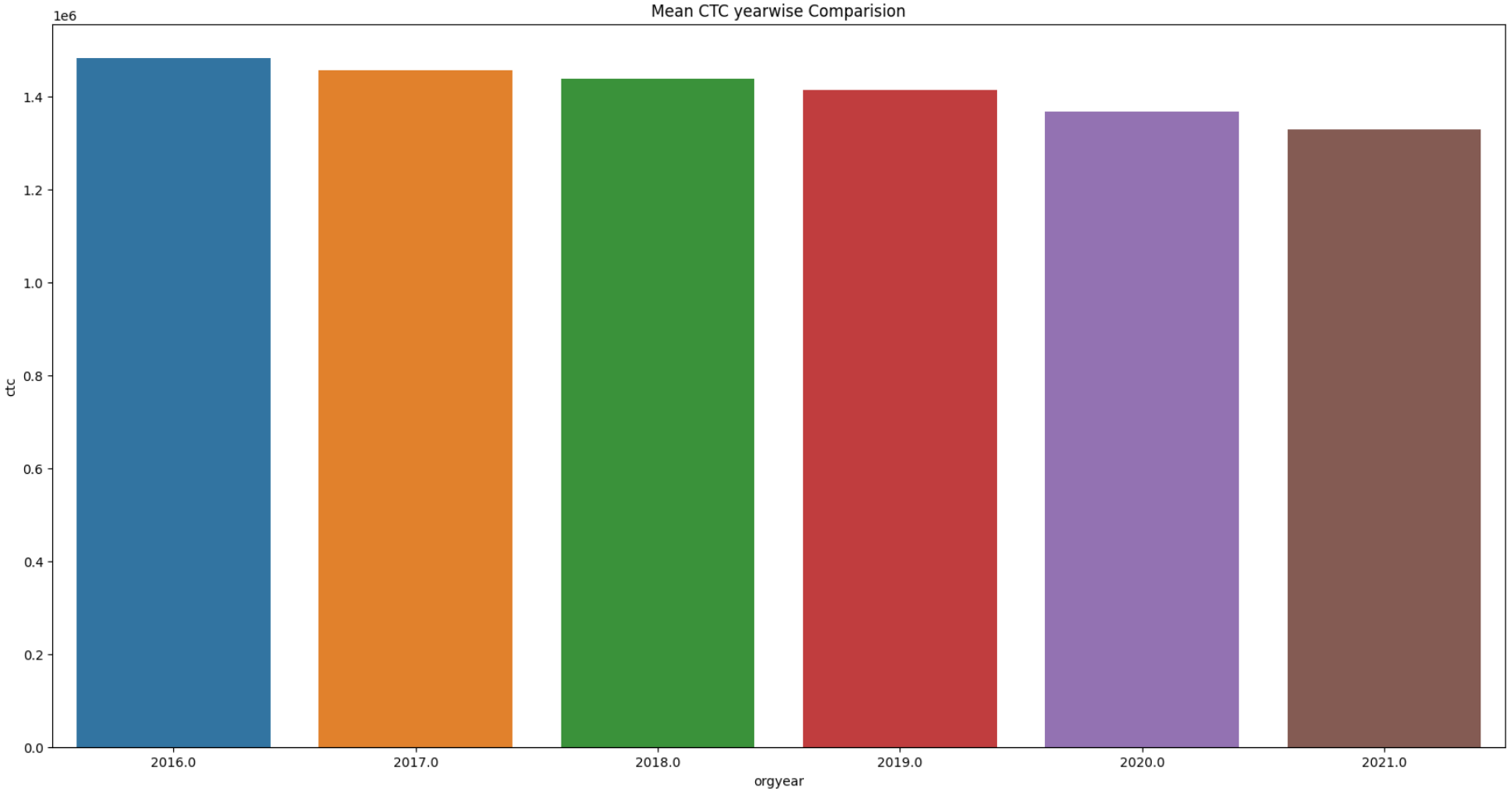


In [47]:

```
tmp = dftmp.copy()

tmp = tmp[tmp['orgyear'] >= 2016]
tmp = tmp.groupby(['orgyear']).mean()['ctc'].reset_index().sort_values('ctc',ascending=False).head(50)
plt.figure(figsize=(20,10))
```

```
sns.barplot(data=tmp,y='ctc',x='orgyear').set(title="Mean CTC yearwise Comparision")
plt.show()
```

```
C:\Users\deepa\AppData\Local\Temp\ipykernel_17216\3323709875.py:4: FutureWarning: The default value of numeric_only in DataFram
eGroupBy.mean is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select onl
y columns which should be valid for the function.
  tmp = tmp.groupby(['orgyear']).mean()['ctc'].reset_index().sort_values('ctc',ascending=False).head(50)
```



Manual Clustering

Creating Designation Flag

In [48]:
```
grp = ['company_hash','job_position','YoE']
data_tmp1 = dateda.groupby(grp).agg({'ctc':['mean','median','min','max','count']}).reset_index()
data_tmp1.columns  = ["{} {}".format(b_, a_) if a_ not in grp else "{}".format(a_) for a_, b_ in zip(data_tmp1.columns.droplev
data_tmp1.head(100).tail(50)

datatmp = dateda.merge(data_tmp1[['company_hash', 'job_position', 'YoE', 'mean ctc']],on=['company_hash', 'job_position', 'YoE



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

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

Creating Class Flag

In [49]:
```
grp = ['company_hash','job_position']
data_tmp1 = datatmp.groupby(grp).agg({'ctc':[('mean2','mean'),'median','min','max','count']}).reset_index()
data_tmp1.columns  = ["{} {}".format(b_, a_) if a_ not in grp else "{}".format(a_) for a_, b_ in zip(data_tmp1.columns.droplev
data_tmp1.head(100).tail(50)


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


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

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

Creating Tier Flag

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


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


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

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

In [51]:
```
datatmp['diff_desig'] = datatmp['ctc'] - datatmp['mean ctc']
datatmp['diff_class'] = datatmp['ctc'] - datatmp['mean2 ctc']
datatmp['diff_tier'] = datatmp['ctc'] - datatmp['mean3 ctc']
```

Based on the manual clustering done so far, answering few questions

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

In [52]:
```
datatmp[datatmp['Tier'] == 1].sort_values('diff_tier',ascending=False).head(10)[['email_hash','ctc','mean3 ctc']]
```

Out[52]:

	email_hash	ctc	mean3 ctc
76180	e15abfd41c005995728191e49ef001e83e813cd3ed5104...	4240000	1.051315e+06
49030	90d5114ca752c55babef2c517ac8b17aaee3d9ff5740de...	4200000	1.051315e+06
59575	b022b84623593cc38a3c1d39d4545b368a7b5f286be1c7...	4200000	1.051315e+06
54761	a1c1c8919e2918b24241a40271e02381daf199c61d7a3b...	4200000	1.143837e+06
70667	d13d7376e9ced16b4e250d0643f9139f8b36a62847f71b...	4200000	1.147773e+06
31649	5d872e52cb535a71fc75a5a97e779bb4c1554d0baa920d...	4200000	1.158025e+06
14808	2b10e1d996c6ab5e175eea35ca25ea7afbaacd1237ab64...	4200000	1.158025e+06
47727	8d0ed00904247626f5557f5983feeb5a0567d7726eea39...	4200000	1.176534e+06
31826	5dff6a65d548553262b6a289f014b2b72a5d47ff6dfa5c...	4170000	1.165011e+06
45627	86b90dd64ddb663ea35be98422947a01ba9ab837fb76df...	4000000	1.051315e+06

Top 10 employees of data science in Amazon / TCS etc earning more than their peers - Class 1

In [53]:
```
datatmp[(datatmp['Tier'] == 1)&(datatmp['Class'] == 1)&(datatmp['job_position'].isin(['Data Science Analyst','Data Scientist',
```

Out[53]:

	email_hash	ctc	mean2 ctc
81289	f04a0228e5af6f8f6ecc33e089892e80d85b3c749b3244...	4000000	1.533750e+06
56231	a63f3f44de7586430615a8a9bd13d41e7b0d541ca0f690...	4200000	1.862000e+06
16846	31616edfc502824631b11793313d35d5bb2288319dcb25...	3800000	1.513842e+06
21441	3efbb8c4d67b4a4c6ba4c639cd84e9ff98b85d5f57d82f...	3979999	1.716000e+06
33512	62f705ba342cb9e51117446a5522c2e42c14db27b9b20e...	4250000	2.025000e+06
83396	f67ae342b7431f7ab05eca998d904647b02711538aa839...	3750000	1.565556e+06
83524	f6e8c41a40ec308c996d498e22729359d2b564cae037a0...	3500000	1.410000e+06
191	009ded427ebcb5c2fb1970017a683693a7abef0fa96f5e...	3900000	1.834333e+06
79529	eb35a5d34977c6135372e46d6cc4f85332f1a4f9578bd5...	4080000	2.020000e+06
36086	6aa8cfeb5b98da66158e0af4ca8869362174abdba84a02...	3200000	1.233235e+06

Bottom 10 employees of data science in Amazon / TCS etc earning less than their peers - Class 3

In [54]:
```
datatmp[(datatmp['Tier'] == 1)&(datatmp['Class'] == 3)&(datatmp['job_position'].isin(['Data Science Analyst','Data Scientist',
```

Out[54]:

	email_hash	ctc	mean2 ctc
14514	2a3136f6e2d03a3dbfa3f683e4ae1b744b4815a8e0177c...	1700000	3125000.0
55794	a4f1770283497277f8cd3b7cb04e9b5c3135815eebb4cf...	2300000	3292500.0
48870	9069f6772b1e7959734a115bf49b2168a888608496af50...	1900000	2850000.0
82770	f49bd18e7fe914929f6cc23bb4e7979d58290119f2adcf...	1600000	2500000.0
51648	987a063524741381c302a096e4b019f46088e519f59f4a...	2000000	2750000.0
65945	c371eff30d6983ab69401441f359fed64397f7699c7aff...	1630000	2350000.0
79574	eb5552cf683e3072a7e2e2c6e63ebb46183a716b2bd2a1...	1780000	2496000.0
2812	080c3b2cc8fe9e7743520a3771a3b4db72e49ef2542ebf...	1400000	1986000.0
26908	4fcbc73fbd3da62f8750d69c13846ada4d1302f4817865...	1700000	2250000.0
61631	b63f00fbd2f8774eccde057bbf3f99ae1742adf496b2cc...	1600000	2102500.0

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

In [55]:
```
datatmp[datatmp['Tier'] == 3].sort_values('diff_tier',ascending=True).head(10)[['email_hash','ctc','mean3 ctc']]
```

Out[55]:

	email_hash	ctc	mean3 ctc
12121	2322345290a1926df62347d45f06b68932e219cb010bf8...	850000	3.262923e+06
64065	bda6e0f742115289a27f304078935331a5563d90c91461...	750000	2.929000e+06
15908	2e7e946b56a245338d8da1daf60ef851031c9964cffd25...	950000	2.950000e+06
4334	0c535bb44414d62cab133425339bd7e156ec79823899ae...	810000	2.770000e+06
73476	d96a6540ff59456abe30f51f68e954388b1f6922c4bb0c...	900000	2.840543e+06
49904	935480e039d80833292d858a553a4bc0f628b9b97ce9ec...	900000	2.840543e+06
19346	38d71a484d7663f7c14df8432620bbbab718933173a295...	1368000	3.262923e+06
70292	d034e386dbce817ee1ea099b161379d3341af0a16573d8...	800000	2.683125e+06
36006	6a6d1a4452505b678e264700fd0c28f247c4522d27f112...	770000	2.637273e+06
2612	077fd3f95d8dbf89c112a8eca6601db3729f51b53b57a0...	720000	2.577054e+06

Top 10 employees in "Qhmqxp Xzw" department - having 5/6/7 years of experience earning more than their peers - Tier X

In [79]:
```
datatmp[(datatmp['YoE'].isin([5,6,7]))&(datatmp['company_hash'].isin(['Qhmqxp Xzw']))].sort_values('diff_desig',ascending=Fals
```

Out[79]:

	email_hash	ctc	mean ctc
46527	897af8080bb684d778f8bc696daf72557f78d34de21ed7...	1200000	1200000.0
54911	a22d8162143d3d170a7659ada8de702534dcfe532ae17c...	3800000	3800000.0
67006	c69263a69e835f6ceee29a67c8a2e97aa2032a29198ec6...	3600000	3600000.0
79486	eb17d547e358233f375901890631eda3b7fbae8574d9e8...	1100000	1100000.0

Top 10 companies (based on their CTC)

In [57]:
```
datatmp.groupby('company_hash').mean()['ctc'].reset_index().sort_values('ctc',ascending=False).head(10)[['company_hash','ctc']
```

```
C:\Users\deepa\AppData\Local\Temp\ipykernel_17216\3415716943.py:1: FutureWarning: The default value of numeric_only in DataFram
eGroupBy.mean is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select onl
y columns which should be valid for the function.
  datatmp.groupby('company_hash').mean()['ctc'].reset_index().sort_values('ctc',ascending=False).head(10)[['company_hash','ct
c']]
```

Out[57]:

	company_hash	ctc
12498	Tqxwoogz Qa Mvzsvrgqt	4250000.0
9063	Ovrnoxat Ntwyzgrgsj	4250000.0
19694	Zvsqv Cxoxgz Xzaxv Uqxcvnt Rxbxnta	4220000.0
6183	Mvqntqerjxg	4200000.0
5527	Lxeert Ntwyzgrgsxto	4200000.0
14758	Vrjgoyv Ntwyzgrgsj Wgzohrnvzwj Otqcxwt	4200000.0
7735	Obvqnmxnuxdtr Ntwy Ucn Rna	4200000.0
9208	Owm	4200000.0
6559	Nhqmgyxqt	4200000.0
3186	Ertwp	4200000.0

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

In [59]:
```
tmp = datatmp[datatmp['job_position'] != 'na']
tmp = tmp.groupby(['company_hash','job_position']).mean().sort_values(['company_hash','ctc']).reset_index()
tmp = tmp.groupby('company_hash').head(2)[['company_hash','job_position']]
tmp
```

```
C:\Users\deepa\AppData\Local\Temp\ipykernel_17216\2740876429.py:2: FutureWarning: The default value of numeric_only in DataFram
eGroupBy.mean is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select onl
y columns which should be valid for the function.
  tmp = tmp.groupby(['company_hash','job_position']).mean().sort_values(['company_hash','ctc']).reset_index()
```

Out[59]:

	company_hash	job_position
0	01 Ojztqsj	Frontend Engineer
1	05Mz Exzytvrny Uqxcvnt Rxbxnta	Backend Engineer
2	1 Jtvq	Backend Engineer
3	10 Axsxnvr Ahmvx Rgzagz	Android Engineer
4	1001 Vuuo	Frontend Engineer
...	...	...
32650	Zxzvzxjv Sqghu	Engineering Leadership
32651	Zyuw Rxbxnta	Frontend Engineer
32652	Zyvzwt Wgzohrnxzs Tzsxzttqo	Frontend Engineer
32653	Zz	Other
32654	Zzzbzb	Other

22950 rows × 2 columns

Top 2 positions in top Paying companies

In [60]:
```
tmp[tmp['company_hash'].isin(['Ojbvzntw',
 'Sgrabvz Ovwyo',
 'Qhmqxp Xzw',
 'Bgqsvz Onvzrtj',
 'Ovrnoxat Ntwyzgrgsj',
 'Wxowg',
 'Ovbohzs Trtwnqgzxwo',
 'Owyztxatq Trtwnqxw Xzaxv',
 'Tqxwoogz Qa Mvzsvrgqt',
 'Uyxrxuo',
 'Xzntr Ntwyzgrgsj Xzaxv Ucn Rna',
 'Vbvkgz',
 'Ktgnvu',
```

```
'Vruyvsqtu Otwhqxnxto',
'Fvrbvqn Rvmo',
'Ojbvzntw Ogenfvqt Ogrhnxgzo',
'Ozvuatvr',
'Zcxaxv',
'Nvnv Wgzohrnvzwj Otqcxwto',
'Ottpxej',
'Amo Mvzp',
'Bxwqgogen',
'St',
'Bgtzsvst',
'Vpvbvx Ntwyzgrgsxto',
'Cbfvqt',
'Rxzptaxz',
'Qvphntz',
'Otqcxwtzgf',
'Zvsqv Cxoxgz Xzaxv Uqxcvnt Rxbxnta',
'Otvqo Ygraxzso Wgqugqvnxgz',
'Nvnv Wgbbhzxwvnxgzo',
'Bvptbjnqxuwgb',
'Nvqstn',
'Ztnvuu',
'Srgmvrrgsxw Xzw',
'Xpohrv',
'Avjngzv Ojontbo',
'Hzvwvatbj',
'Rgctmgzxng',
'Gqvwrt',
'Avcxznv',
'Hxuvny',
'Ngftq Qtotvqwy Wvuxnvr',
'Mn',
'Uvjnb',
'Srgmvrexqb Rxbxnta',
'Yhvftx Ntwyzgrgsxto',
'Zvnxgzvr Ongwp Tdwyvzst Ge Xzaxv Rxbxnta',
'Whqtexn'])]
```

Out[60]:

	company_hash	job_position
477	Amo Mvzp	Data Analyst
478	Amo Mvzp	SDET
1052	Avcxznv	Backend Architect
1063	Avjngzv Ojontbo	Backend Engineer
1783	Bgqsvz Onvzrtj	Associate Software developer
...	...	...
32056	Ztnvuu	Professional Services Engineer
32057	Ztnvuu	Support Engineer
32257	Zvnxgzvr Ongwp Tdwyvzst Ge Xzaxv Rxbxnta	Backend Engineer
32258	Zvnxgzvr Ongwp Tdwyvzst Ge Xzaxv Rxbxnta	Backend Architect
32325	Zvsqv Cxoxgz Xzaxv Uqxcvnt Rxbxnta	Engineering Leadership

91 rows × 2 columns

Preparing data for training model(Imputation/Scaling)

In [61]:
```
data = dateda.copy()
data
```

Out[61]:

	email_hash	Unnamed: 0	company_hash	orgyear	ctc	job_position	ctc_updated_year
0	00003288036a44374976948c327f246fdbdf0778546904...	84782	Bxwqgogen	2012.0	3500000	Backend Engineer	2019.0
3	000120d0c8aa304fcf12ab4b85e21feb80a342cfea03d4...	53905	Bxwqgotbx Wgqugqvnxgz	2004.0	2000000	FullStack Engineer	2021.0
4	00014d71a389170e668ba96ae8e1f9d991591acc899025...	138707	Fvrbvqn Rvmo	2009.0	3400000	na	2018.0
6	00022dc29c7f77032275182b883d4f273ea1007aefc437...	7782	Xzeqvwrgha Ntwyzgrgsxto	2016.0	750000	Frontend Engineer	2019.0
7	00036c2c5212d88d07acdc5bda7eef5653f8b09bbe30b7...	30543	Ocu Xnivz Gbvz	2011.0	2300000	Other	2021.0
...	...	...	...	...	...	...	...
153432	fffa3a7b849802580a1972f11d192b43ff1c871bb43002...	79890	Nvnv Wgzohrnvzwj Otqcxwto	2014.0	1800000	Backend Engineer	2019.0
153438	fffc254e627e4bd1bc0ed7f01f9aebbba7c3cc56ac914e...	39683	Tqxwoogz Ogenfvqt Wvbuho	2004.0	3529999	QA Engineer	2019.0
153439	fffcf97db1e9c13898f4eb4cd1c2fe862358480e104535...	186656	Trnqvcg	2015.0	1600000	na	2018.0
153440	fffe7552892f8ca5fb8647d49ca805b72ea0e9538b6b01...	148878	Znn Avnv Srgmvr Atrxctqj Otqcxwto	2014.0	900000	Devops Engineer	2019.0
153442	ffffa3eb3575f43b86d986911463dce7bcadcea227e5a4...	117170	Sgrabvz Ovwyo	2018.0	1500000	FullStack Engineer	2021.0

86464 rows × 14 columns

Transforming ctc feature using log function

In [62]:
```
data['ctc_log'] = np.log2(data['ctc'])
```

Columns like ['job_position','email_hash','Unnamed: 0','company'] are text.
We can't use them during imputation, so we'll remove these columns

In [63]:
```
drop_cols = ['job_position','email_hash','Unnamed: 0','company_hash']
for i in drop_cols:
    try:
        data.drop([i],axis=1,inplace=True)
    except:
        print('no')
```

In [64]:
```
data.columns
```

Out[64]:
```
Index(['orgyear', 'ctc', 'ctc_updated_year', 'orgyear_na',
       'ctc_updated_year_na', 'company_hash_na', 'job_position_na', 'YoE',
       'company_hash_encode', 'job_position_encode', 'ctc_log'],
      dtype='object')
```

In [65]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 86464 entries, 0 to 153442
Data columns (total 11 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   orgyear              86424 non-null  float64
 1   ctc                  86464 non-null  int64  
 2   ctc_updated_year     86464 non-null  float64
 3   orgyear_na           86464 non-null  bool   
 4   ctc_updated_year_na  86464 non-null  bool   
 5   company_hash_na      86464 non-null  bool   
 6   job_position_na      86464 non-null  bool   
 7   YoE                  86424 non-null  float64
 8   company_hash_encode  86464 non-null  float64
 9   job_position_encode  86464 non-null  float64
 10  ctc_log              86464 non-null  float64
dtypes: bool(4), float64(6), int64(1)
memory usage: 7.6 MB
```

Summary Statistics

In [66]:
```
data.describe()
```

Out[66]:

	orgyear	ctc	ctc_updated_year	YoE	company_hash_encode	job_position_encode	ctc_log
count	86424.000000	8.646400e+04	86464.000000	86424.000000	86464.000000	86464.000000	86464.000000
mean	2013.804626	1.626541e+06	2019.441432	5.636906	0.002635	1209.710671	20.474913
std	4.354664	8.080584e+05	1.283791	4.225522	0.005558	878.503304	0.662649
min	1970.000000	7.040000e+05	2015.000000	0.000000	0.000007	0.065171	19.425216
25%	2012.000000	1.000000e+06	2019.000000	3.000000	0.000033	319.141310	19.931569
50%	2015.000000	1.400000e+06	2019.000000	5.000000	0.000371	1318.013855	20.416995
75%	2017.000000	2.000000e+06	2020.000000	8.000000	0.002170	2431.717315	20.931569
max	2021.000000	4.250000e+06	2021.000000	51.000000	0.034221	2431.717315	22.019031

In [67]:
```
data.isna().sum()
```

Out[67]:
```
orgyear                40
ctc                     0
ctc_updated_year        0
orgyear_na              0
ctc_updated_year_na     0
company_hash_na         0
job_position_na         0
YoE                    40
company_hash_encode     0
job_position_encode     0
ctc_log                 0
dtype: int64
```

Training Model

In [68]:
```
from sklearn.impute import KNNImputer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.cluster import MiniBatchKMeans, KMeans
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
```

Kmeans clustering

So we will be training a model with unscaled features too.

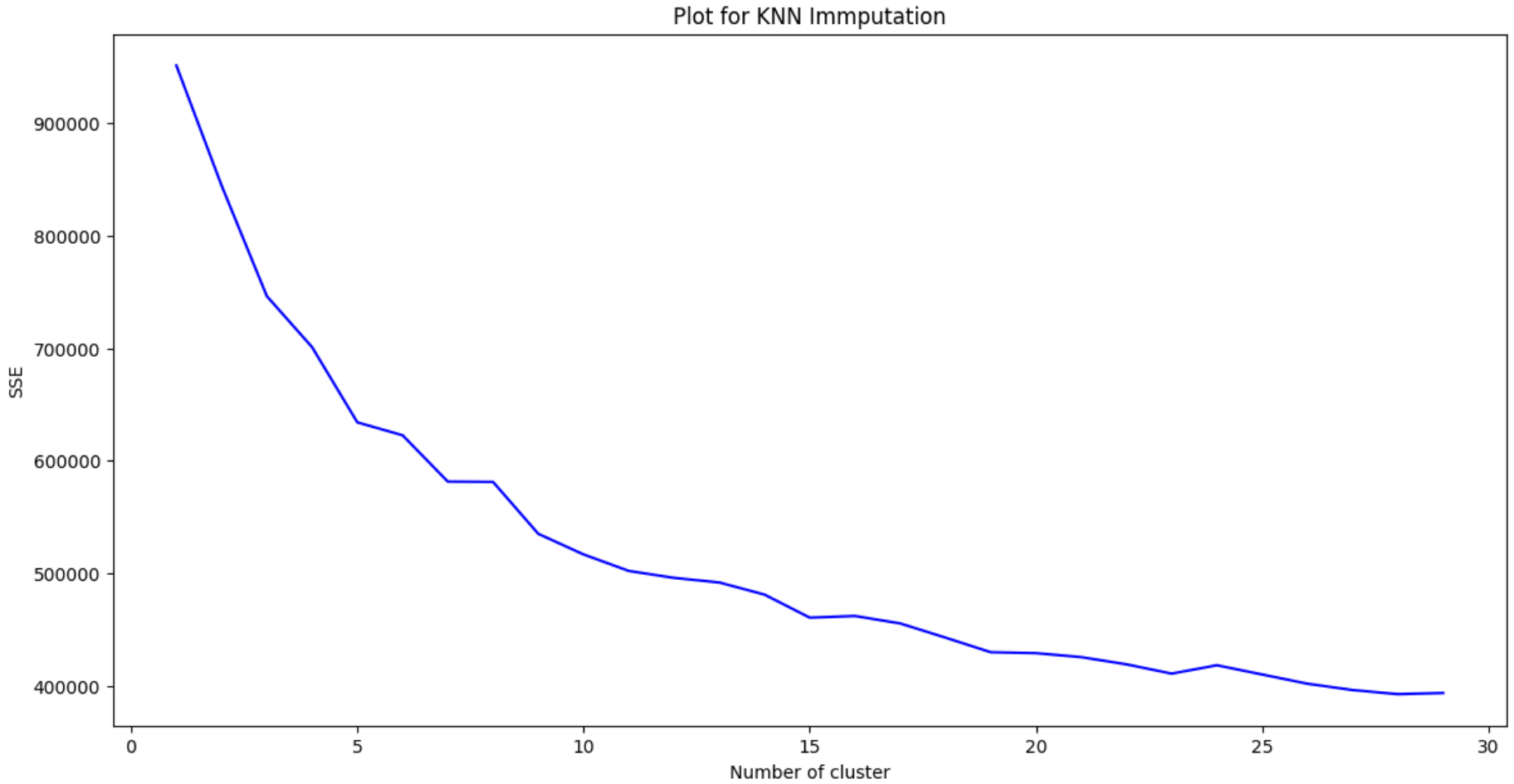
In [69]:
```
pipe_knn = Pipeline([('scaler', StandardScaler()), ('knn_imputer',  KNNImputer(n_neighbors=2, weights="uniform"))])
pipe_knn_5 = Pipeline([('scaler', StandardScaler()), ('knn_imputer',  KNNImputer(n_neighbors=5, weights="uniform"))])
pipe = Pipeline([('scaler', StandardScaler()), ('simple_imputer',  SimpleImputer(missing_values=np.nan, strategy='mean'))])
pipe_knn_pca = Pipeline([('scaler', StandardScaler()), ('knn_imputer',  KNNImputer(n_neighbors=2, weights="uniform")),('pca',P
pipe_unscaled = Pipeline([('knn_imputer',  KNNImputer(n_neighbors=5, weights="uniform"))])
```

Finding optimal num of clusters using Elbow method

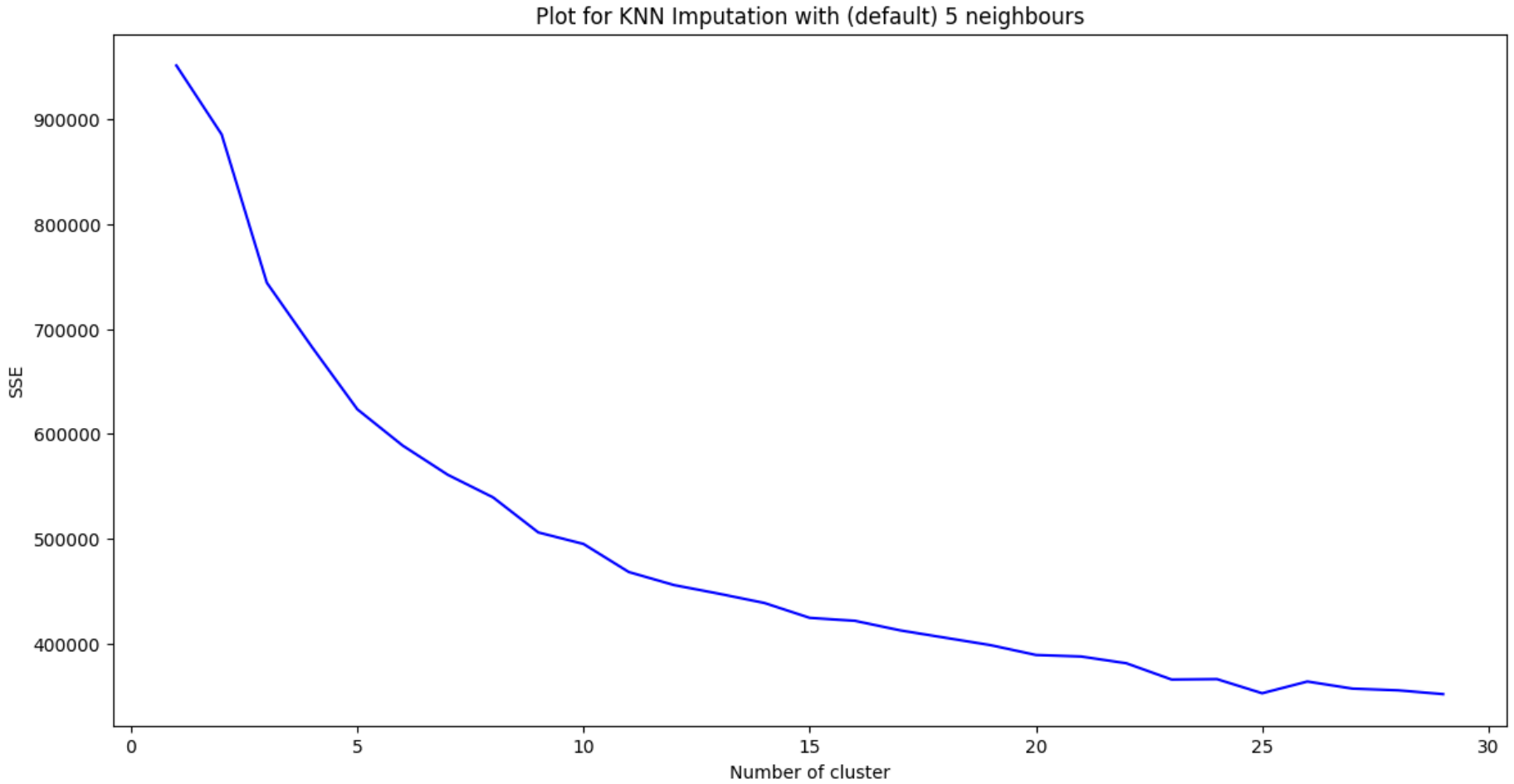
In [82]:
```
for name,pipeline in [('KNN Immputation',pipe_knn),('KNN Imputation with (default) 5 neighbours',pipe_knn_5),('Mean Imputation
                      ('KNN Immputation + PCA', pipe_knn_pca),('KNN Imputation Unscaled data',pipe_unscaled )]:

    X = pipeline.fit_transform(data)
    X = pd.DataFrame(X)
    if "PCA" not in name :
        X.columns= data.columns

    sse = {}
    #sil_score = {}
    print("Running for ",name)
    for k in range(1, 30):
        #print('K :',k)
```

```
        kmeans = MiniBatchKMeans(init="k-means++",n_clusters=k,n_init=3,
                                 random_state=0).fit(X)
        label = kmeans.labels_
        data["clusters"] = label
        #print(data["clusters"])
        sse[k] = kmeans.inertia_

        #sil_score[k] = silhouette_score(X, label, metric='euclidean')

    plt.figure(figsize=(14,7))
    plt.plot(list(sse.keys()), list(sse.values()),'b-',label='Sum of squared error')
    plt.xlabel("Number of cluster")
    plt.ylabel("SSE")
    plt.title("Plot for "+name)
    plt.show()
```

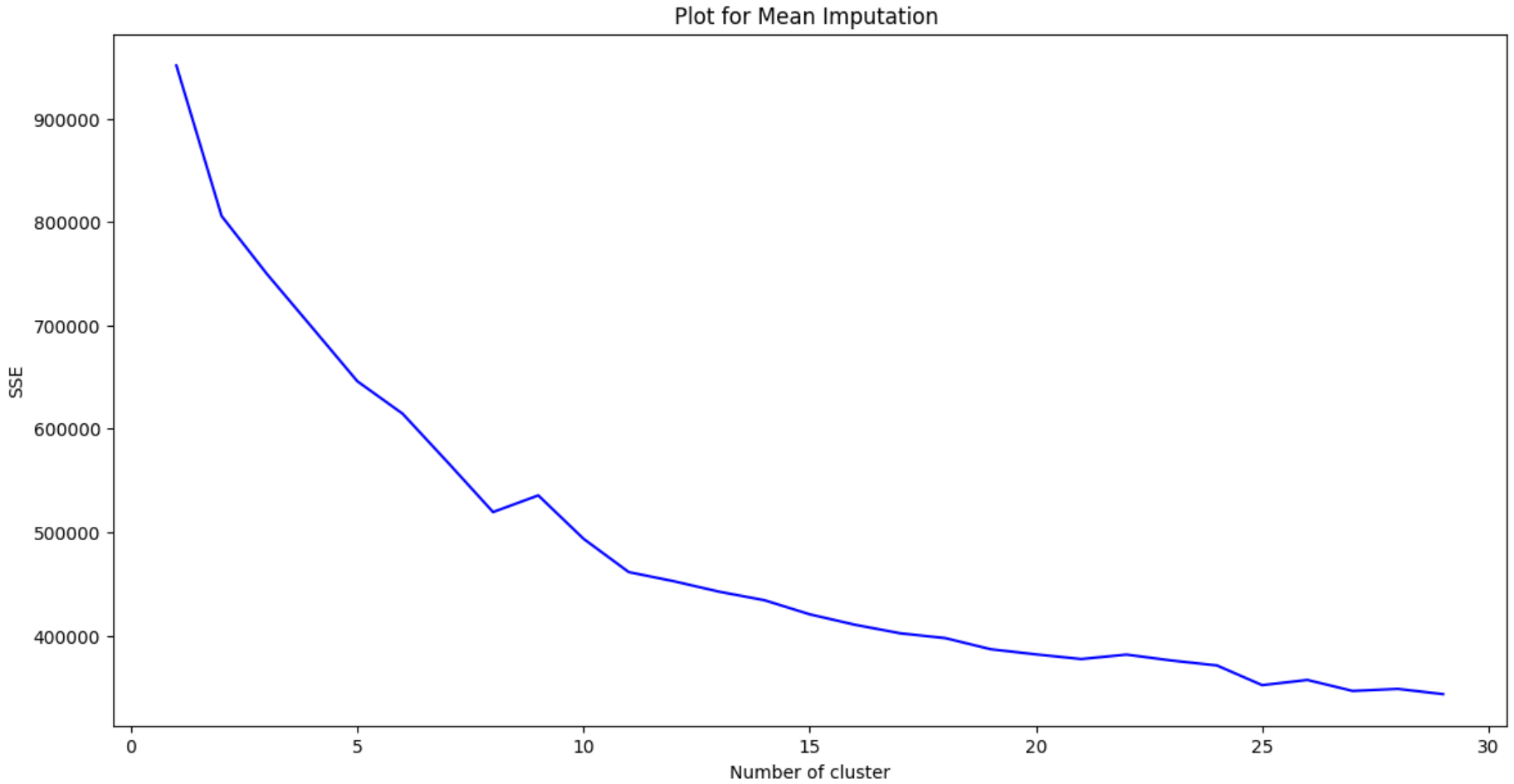
```
Running for  KNN Immputation
```



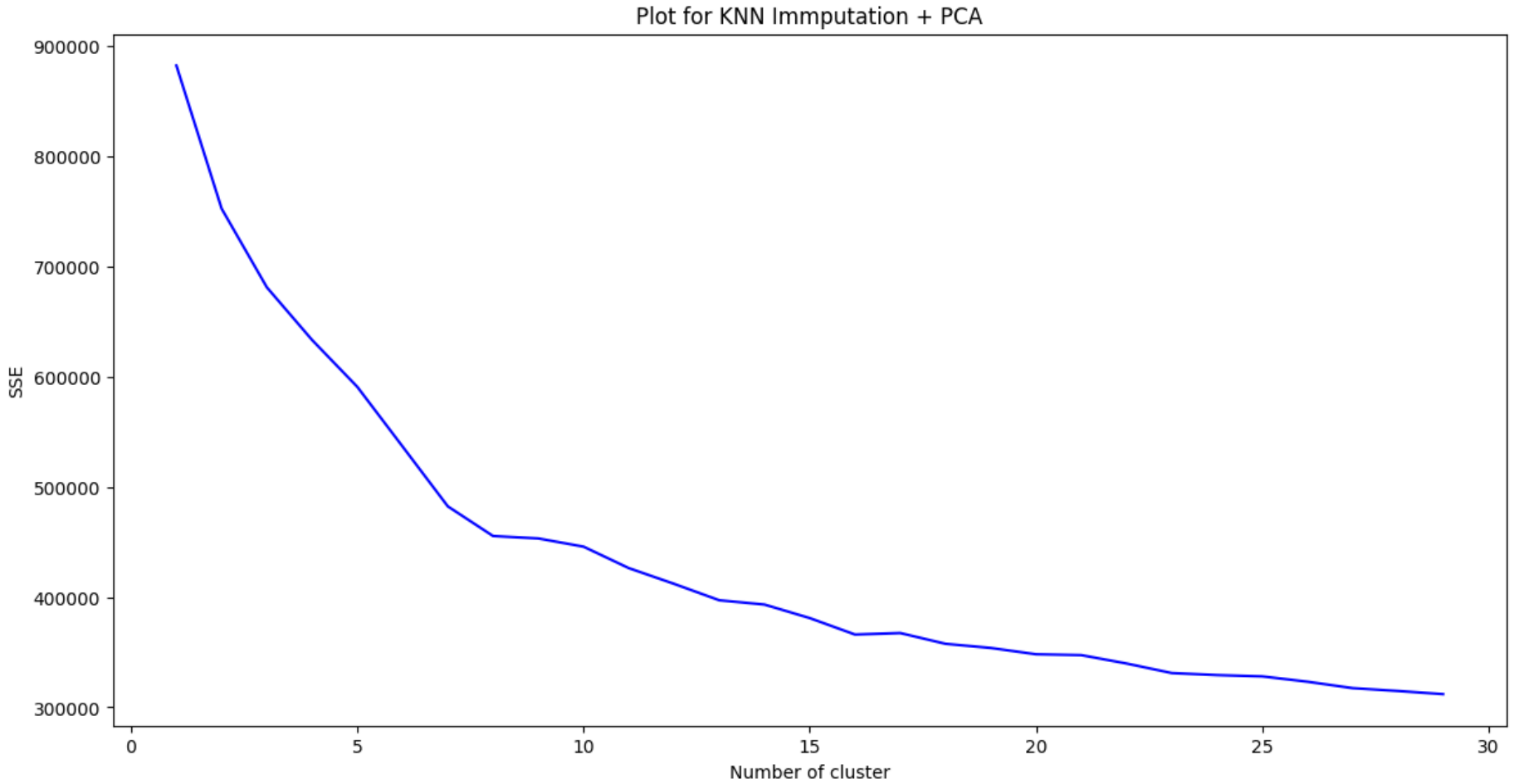

```
Running for  KNN Imputation with (default) 5 neighbours
```



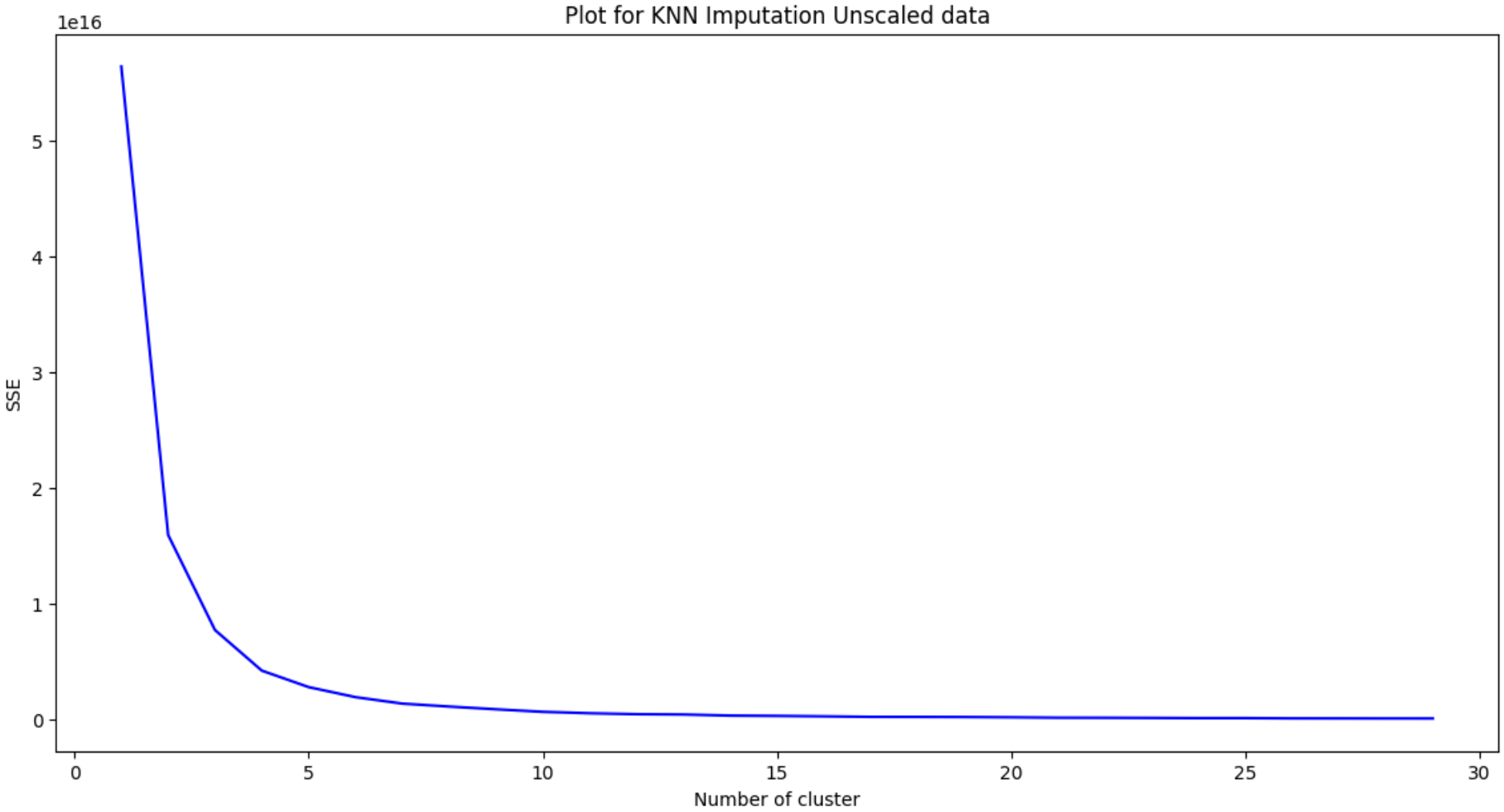

```
Running for  Mean Imputation
```




```
Running for  KNN Immputation + PCA
```




```
Running for  KNN Imputation Unscaled data
```



Insights

Model	n_clusters
KNN Immputation	16
KNN Imputation with (default) 5 neighbours	20

Mean Imputation	25
KNN Immputation + PCA	21
KNN Imputation Unscaled data	5

Number of clusters is around 16-20 for scaled data, while around 5 for unscaled data

In [71]:
```
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram
```

Agglomerative Clustering

In [72]:
```
tmp = X.sample(frac=0.2)

tmp.shape
```

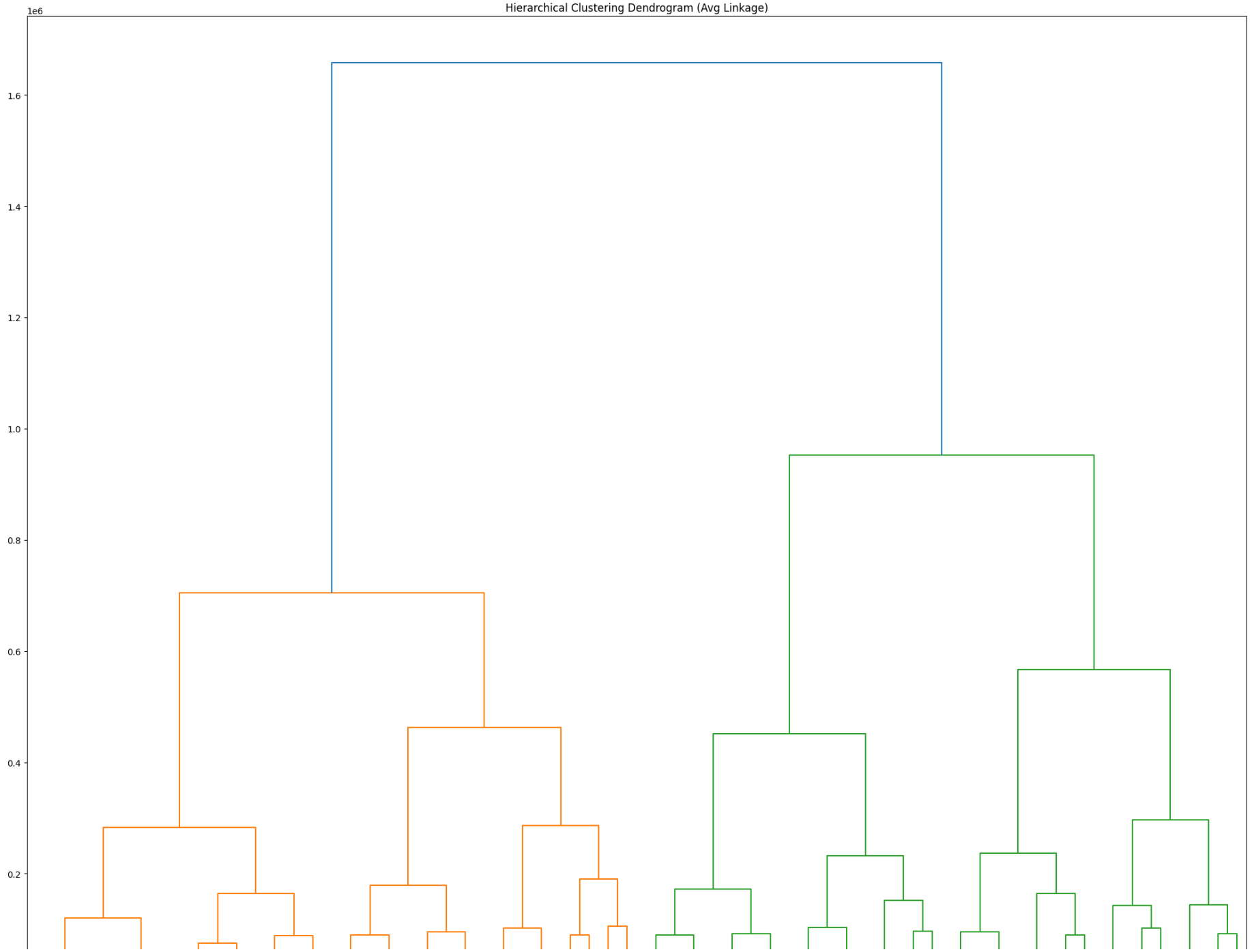
Out[72]: (17293, 12)

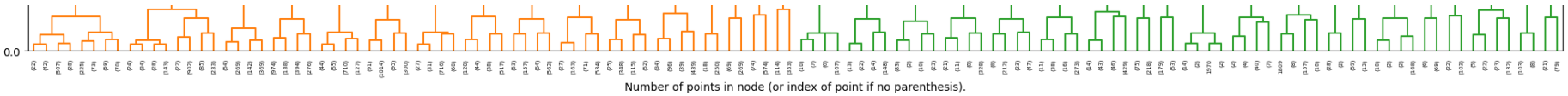
In [73]:
```
def plot_dendrogram(model, **kwargs):
    # Create linkage matrix and then plot the dendrogram

    # create the counts of samples under each node
    counts = np.zeros(model.children_.shape[0])
    n_samples = len(model.labels_)
    for i, merge in enumerate(model.children_):
        current_count = 0
        for child_idx in merge:
            if child_idx < n_samples:
                current_count += 1  # leaf node
            else:
                current_count += counts[child_idx - n_samples]
```

```
        counts[i] = current_count

    linkage_matrix = np.column_stack(
        [model.children_, model.distances_, counts]
    ).astype(float)

    # Plot the corresponding dendrogram

    dendrogram(linkage_matrix, **kwargs)
```

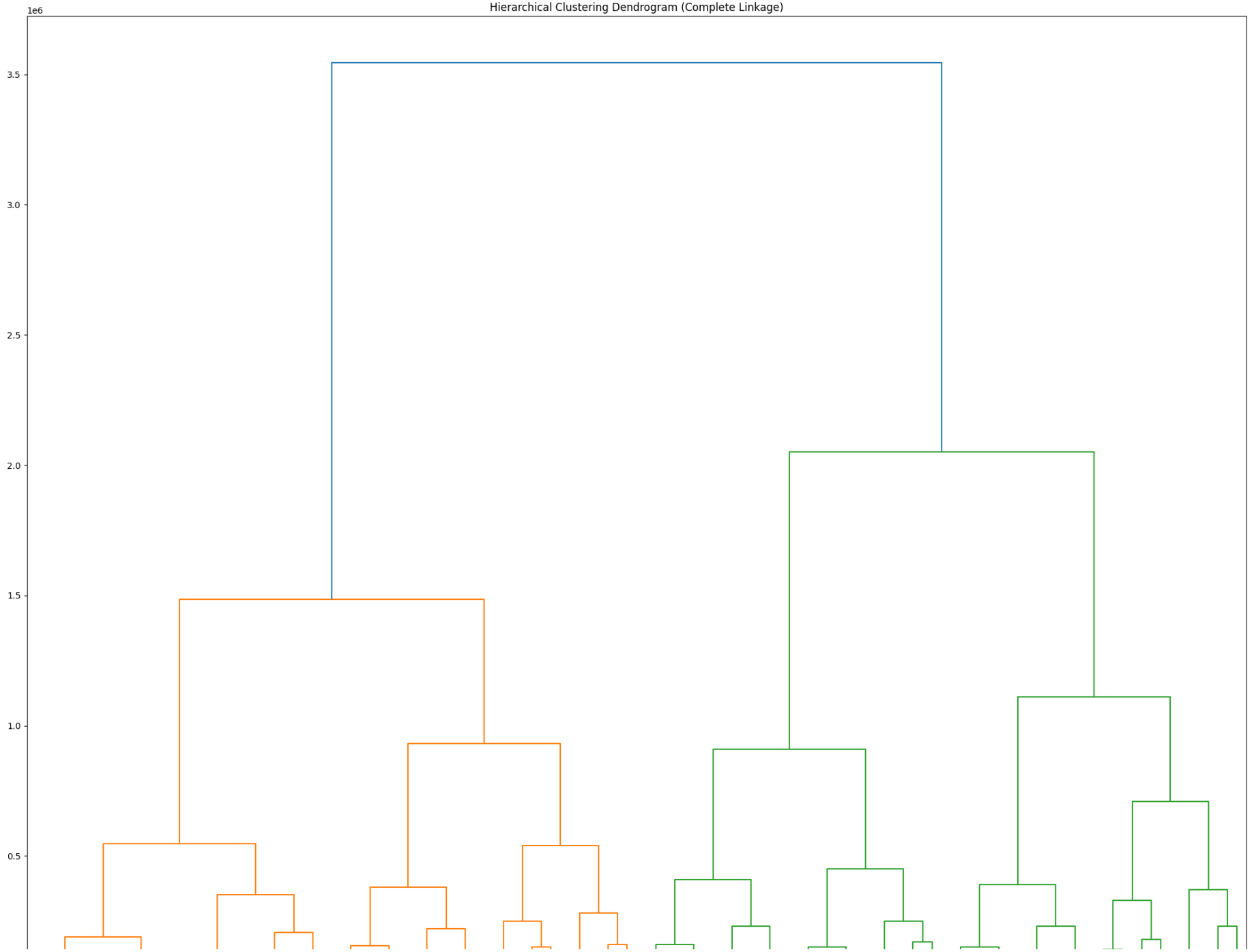
In [74]:

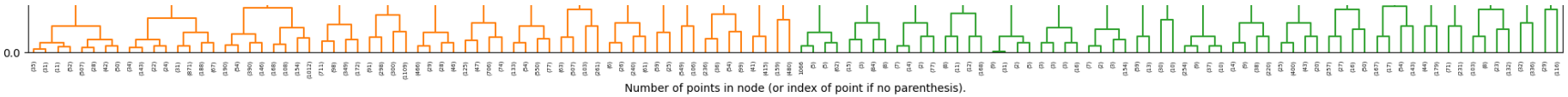
```
model = AgglomerativeClustering(distance_threshold =0, n_clusters=None, compute_distances=True,linkage='average').fit(tmp)

plt.figure(figsize=(25,20))
plt.title("Hierarchical Clustering Dendrogram (Avg Linkage)")
plot_dendrogram(model, truncate_mode="level", p=6)
plt.xlabel("Number of points in node (or index of point if no parenthesis).")
plt.show()
```


Hierarchical Clustering Dendrogram (Avg Linkage)
1e6
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2

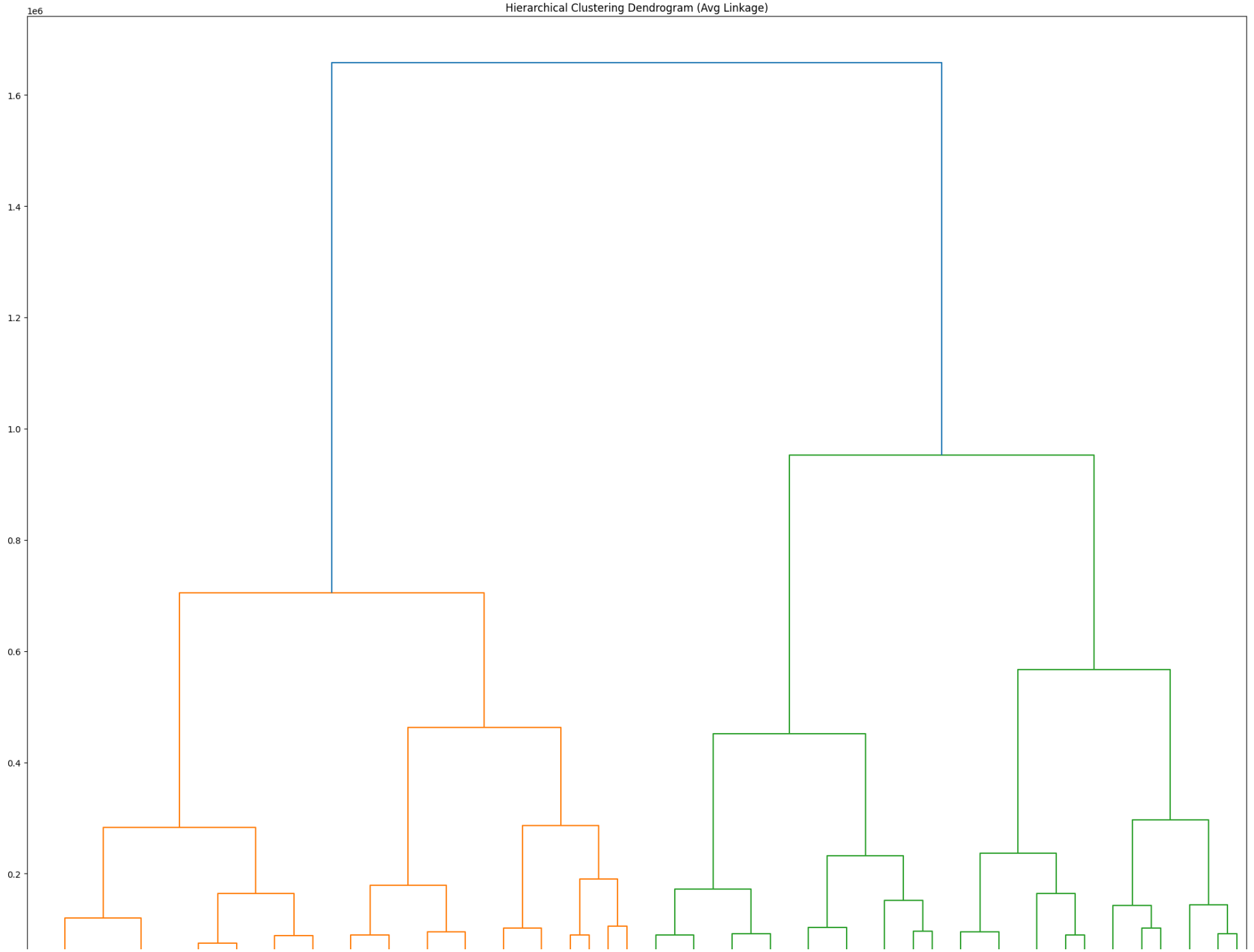


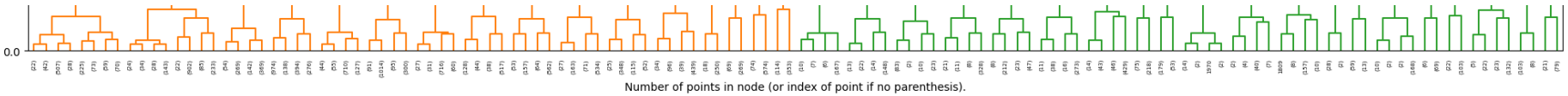
In [75]:
```
model = AgglomerativeClustering(distance_threshold =0, n_clusters=None, compute_distances=True,linkage='complete').fit(tmp)

plt.figure(figsize=(25,20))
plt.title("Hierarchical Clustering Dendrogram (Complete Linkage)")
plot_dendrogram(model, truncate_mode="level", p=6)
plt.xlabel("Number of points in node (or index of point if no parenthesis).")
plt.show()
```


Hierarchical Clustering Dendrogram (Complete Linkage)
1e6
3.5
3.0
2.5
2.0
1.5
1.0
0.5

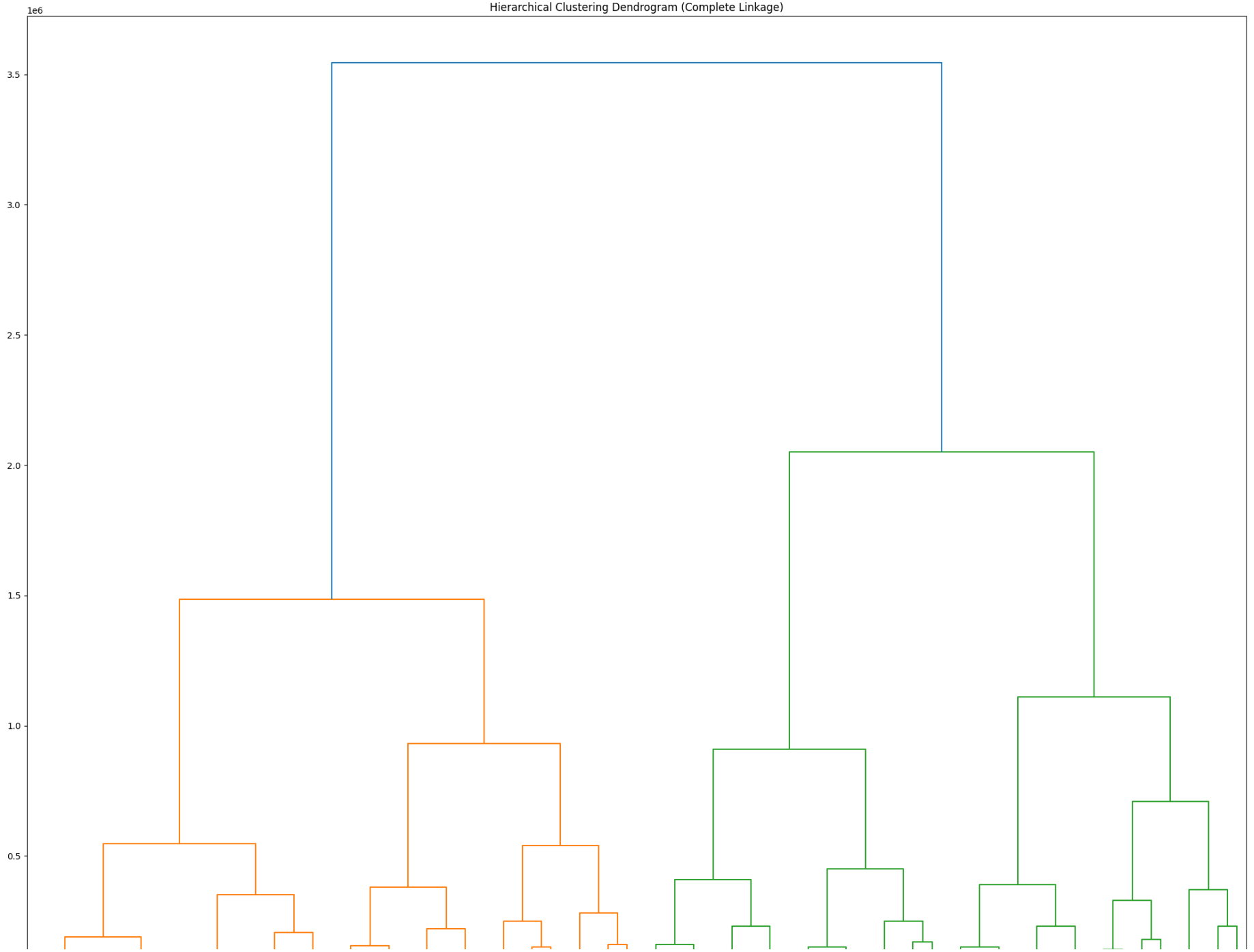


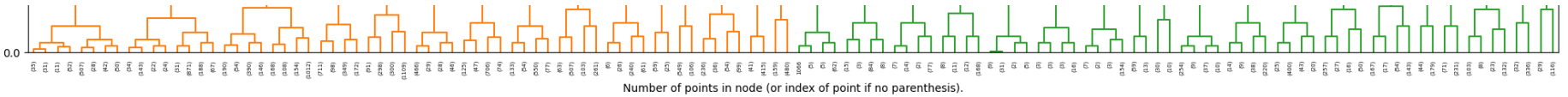
In [76]:
```
model = AgglomerativeClustering(n_clusters=17, compute_distances=True,linkage='average').fit(tmp)

plt.figure(figsize=(25,20))
plt.title("Hierarchical Clustering Dendrogram (Avg Linkage)")
plot_dendrogram(model, truncate_mode="level", p=6)
plt.xlabel("Number of points in node (or index of point if no parenthesis).")
plt.show()
```


Hierarchical Clustering Dendrogram (Avg Linkage)
1e6
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2

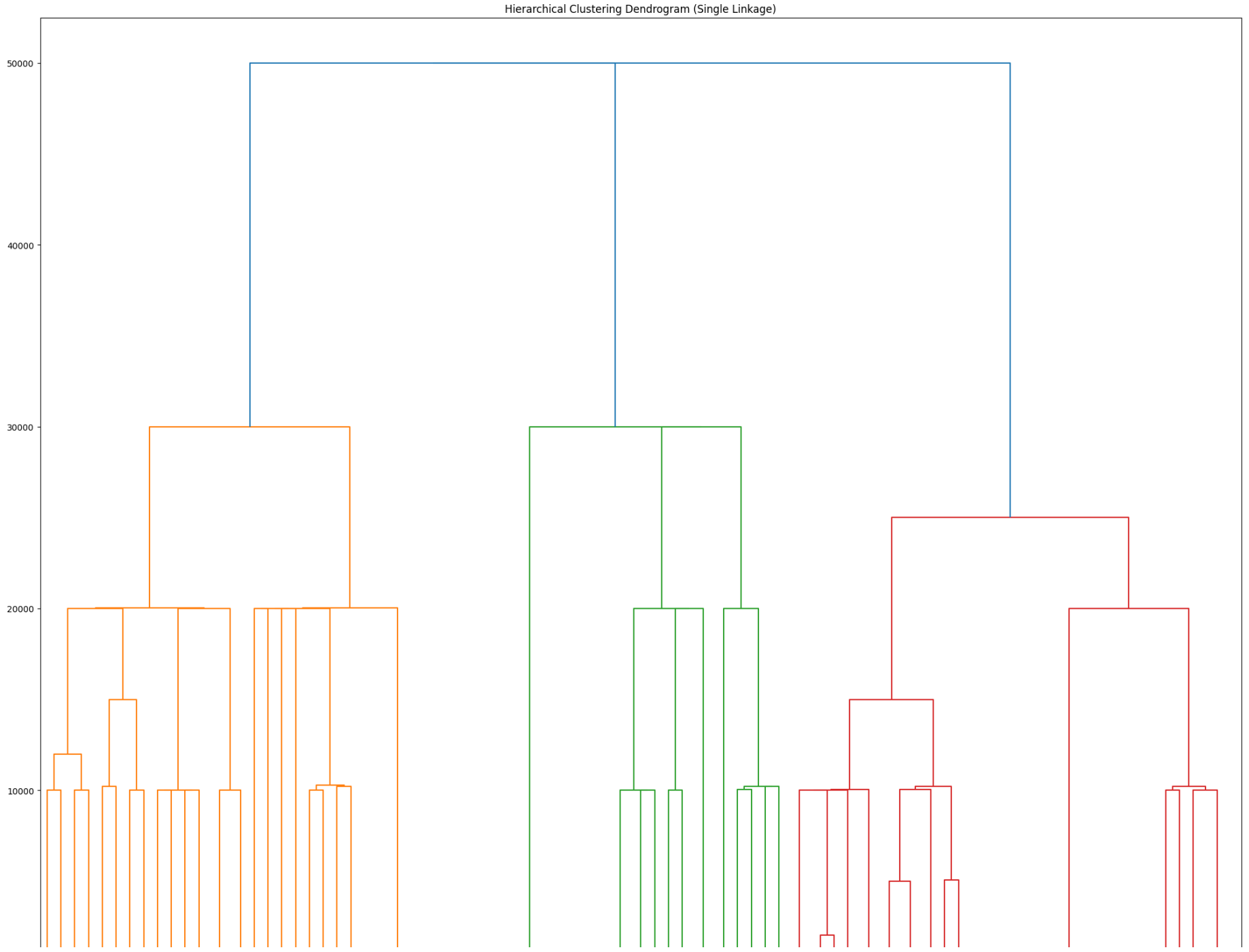


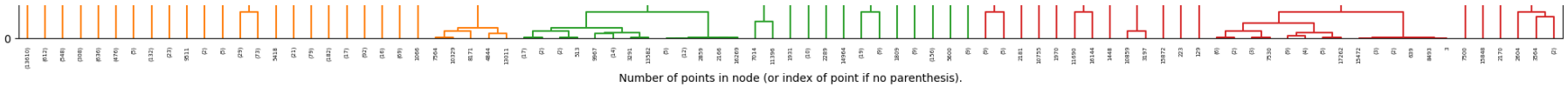
In [77]:
```
model = AgglomerativeClustering(n_clusters=17, compute_distances=True,linkage='complete').fit(tmp)

plt.figure(figsize=(25,20))
plt.title("Hierarchical Clustering Dendrogram (Complete Linkage)")
plot_dendrogram(model, truncate_mode="level", p=6)
plt.xlabel("Number of points in node (or index of point if no parenthesis).")
plt.show()
```


Hierarchical Clustering Dendrogram (Complete Linkage)
1e6
3.5
3.0
2.5
2.0
1.5
1.0
0.5



In [78]:
```
model = AgglomerativeClustering(n_clusters=5, compute_distances=True,linkage='single').fit(tmp)

plt.figure(figsize=(25,20))
plt.title("Hierarchical Clustering Dendrogram (Single Linkage)")
plot_dendrogram(model, truncate_mode="level", p=6)
plt.xlabel("Number of points in node (or index of point if no parenthesis).")
plt.show()
```


Hierarchical Clustering Dendrogram (Single Linkage)
50000
40000
30000
20000
10000



Insights

Number of clusters around 2 seems optimal in most cases, while in last plot(with single linkage) number of clusters around 16 is optimal

Insights/ Recommendations

Insights

- Top Paying job titles include 'Engineering Leadership', 'Backend Engineer', 'Product Manager', 'Program Manager', 'SDET', 'QA Engineer', 'Data Scientist', 'Android Engineer' and 'FullStack Engineer'.
- Top paying companies have been identified.
- Among top paying companies, salary for few organisations is getting lesser in recent years which have been identified and listed in the notebook.
- Among Top paying companies for few companies mean salary is increasing every year, the same have been identified and listed above
- Avg CTC seems to be decreasing with year.

Recommendations

- Freshers who want to work on technical side should look for roles related to Backend Engineer, SDET, QA engineer, Dataa Scientist, Android Engineer,Full stack engineer to get good salaries as expirience increases.
- Freshers who want best CTC should aim for companies identified with higher salaries and according to their tech orientation

In [ ]: