Problem Statement

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

In [1]:

```
import numpy as np
import pandas as pd
import re
import matplotlib.pyplot as plt
import seaborn
from sklearn.preprocessing import MinMaxScaler
```

In [2]:

```
df = pd.read_csv('scaler_clustering.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	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 [4]:

```
df.shape
```

Out[4]:

```
(205843, 7)
```

In [5]:

```
df=df.drop(columns='Unnamed: 0')
df=df.drop(columns='email_hash')
```

In [6]:

```
df.info()
```

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

In [7]:

```
df.describe()
```

Out[7]:

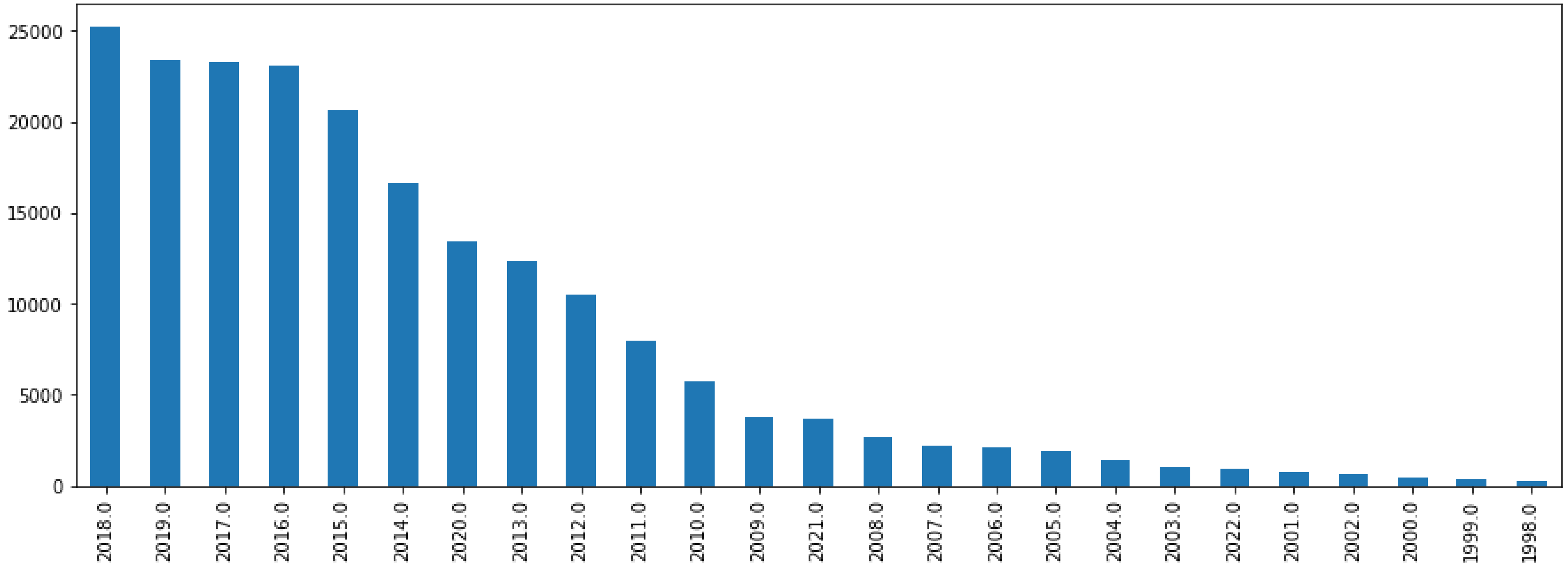
	orgyear	ctc	ctc_updated_year
count	205757.000000	2.058430e+05	205843.000000
mean	2014.882750	2.271685e+06	2019.628231
std	63.571115	1.180091e+07	1.325104
min	0.000000	2.000000e+00	2015.000000
25%	2013.000000	5.300000e+05	2019.000000
50%	2016.000000	9.500000e+05	2020.000000
75%	2018.000000	1.700000e+06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

```
In [8]: df.isnull().sum()/len(df)*100
```

```
Out[8]: company_hash          0.021376
        orgyear               0.041779
        ctc                   0.000000
        job_position         25.534995
        ctc_updated_year      0.000000
        dtype: float64
```

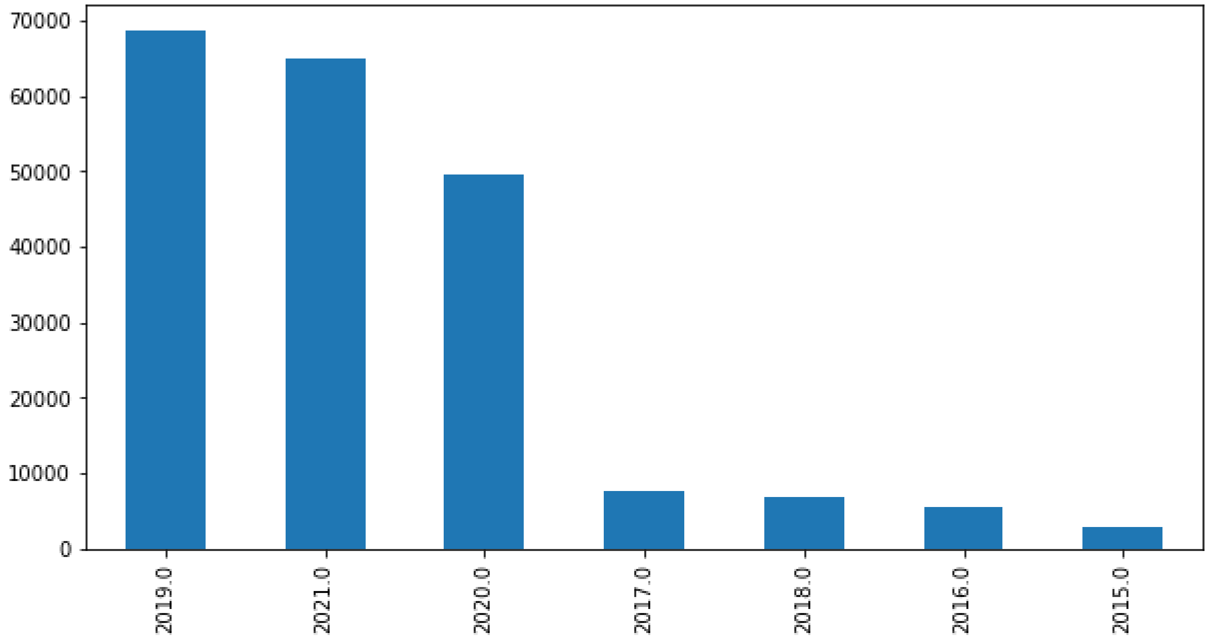
```
In [9]: fig = plt.figure(figsize = (15, 5))
        df['orgyear'].value_counts().head(25).plot(kind='bar')
```

```
Out[9]: <AxesSubplot:>
```



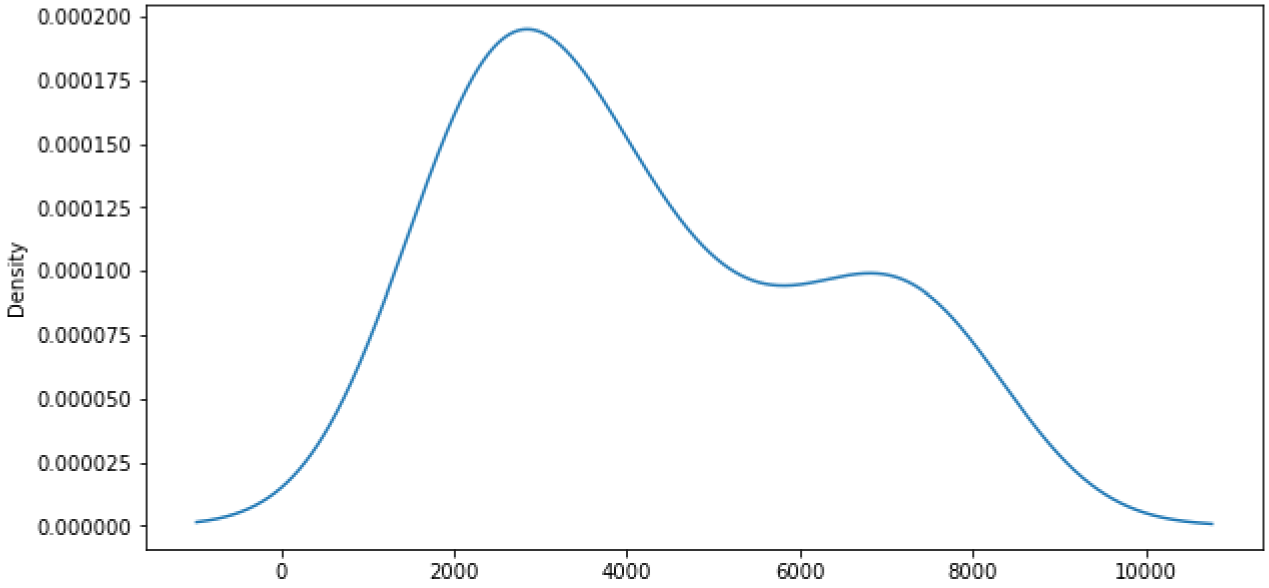
In [10]:
```
fig = plt.figure(figsize = (10, 5))
df['ctc_updated_year'].value_counts().plot(kind='bar')
```

Out[10]:
```
<AxesSubplot:>
```



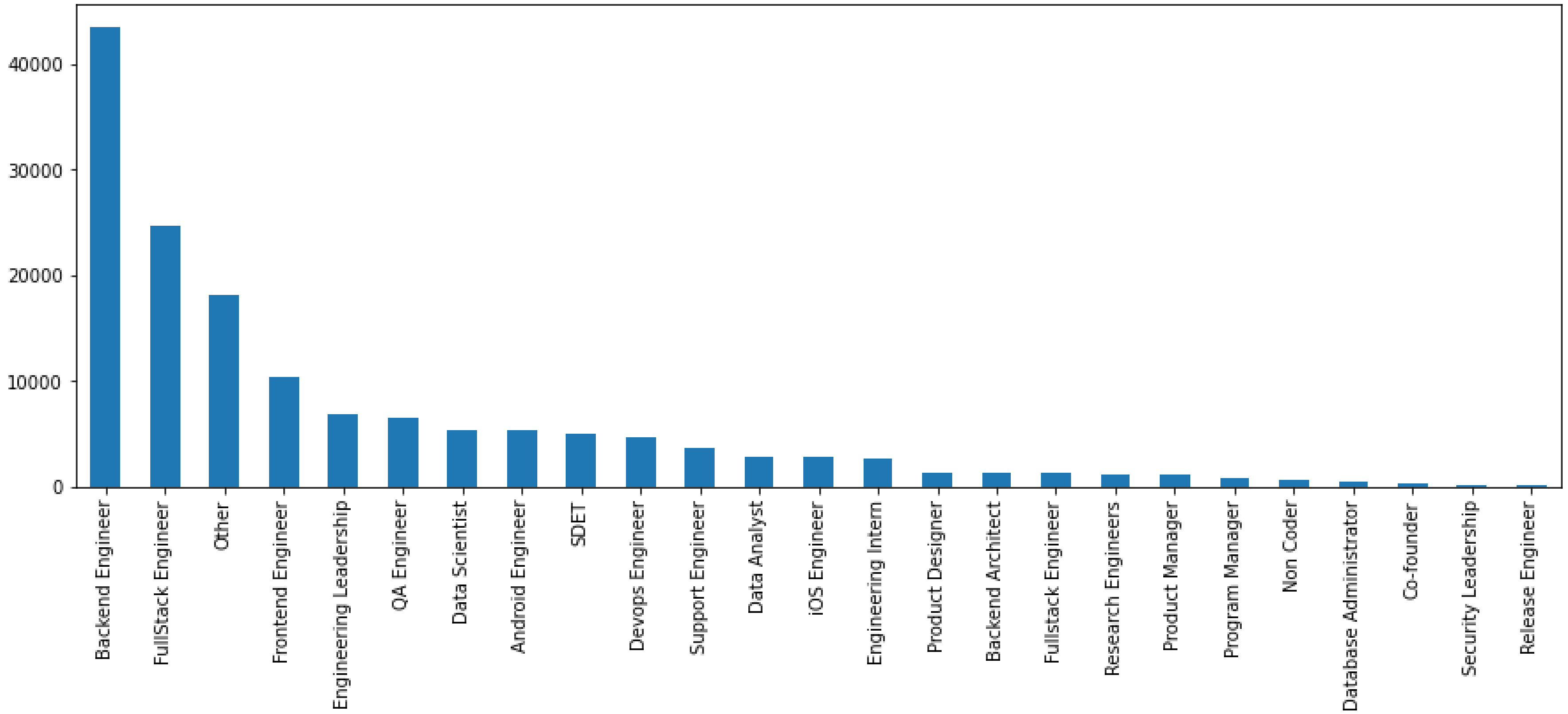
In [11]:
```
fig = plt.figure(figsize = (10, 5))
df['ctc'].value_counts().head(25).plot(kind='kde')
```

Out[11]:
```
<AxesSubplot:ylabel='Density'>
```



In [12]:
```
fig = plt.figure(figsize = (15, 5))
df['job_position'].value_counts().head(25).plot(kind='bar')
```

Out[12]:
```
<AxesSubplot:>
```



Data Pre-processing

In [13]:
```
_data_nums=df.orgyear
#keeping only the numerical columns
```

In [14]:
```
_data_nums=_data_nums.reset_index()
```

In [15]:
```
_data_nums.drop(columns='index',inplace=True)
```

In [16]:
```
columns=_data_nums.columns
```

In [17]:
```
from sklearn.impute import KNNImputer
imputer = KNNImputer(n_neighbors=5, weights='uniform', metric='nan_euclidean',)
imputer.fit(_data_nums)
# transform the dataset
_data_new = imputer.transform(_data_nums)
```

In [18]:
```
_data_new=pd.DataFrame(_data_new)
```

In [19]:
```
_data_new.columns=columns
```

In [20]:
```
_data_new.isnull().sum()
```

Out[20]:
```
orgyear    0
dtype: int64
```

Getting the remaining columns back

In [21]:
```
remaining_columns=list(set(df.columns).difference(set(columns)))
data=pd.concat([_data_new, df[remaining_columns]],axis=1)
data=pd.DataFrame(data)
```

In [22]:
```
data.dropna(inplace=True)
```

Regex for cleaning company names

In [23]:

```
try:
    for i in range(len(data)):
        data['company_hash'][i]=re.sub('[^A-Za-z0-9 ]+', '', data['company_hash'][i])
except Exception:
    pass
```

```
/var/folders/2m/svsbyfss4h53t29lk1vcnvf40000gn/T/ipykernel_7376/1537211381.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#r
eturning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returnin
g-a-view-versus-a-copy)
  data['company_hash'][i]=re.sub('[^A-Za-z0-9 ]+', '', data['company_hash'][i])
```

New Feature Creation

In [24]:

```
data['years_of_Experience']=2023-data['orgyear']

data = data[(data['years_of_Experience'] >= 0) & (data['years_of_Experience'] <= 100)]
```

In [25]:

```
data
```

Out[25]:

	orgyear	job_position	company_hash	ctc_updated_year	ctc	years_of_Experience
0	2016.0	Other	atrgxnnt xzaxv	2020.0	1100000	7.0
1	2018.0	FullStack Engineer	qtrxvzwt xzegwgbb rxbxnta	2019.0	449999	5.0
2	2015.0	Backend Engineer	ojzwnvwnxw vx	2020.0	2000000	8.0
3	2017.0	Backend Engineer	ngpgutaxv	2019.0	700000	6.0
4	2017.0	FullStack Engineer	qxen sqghu	2019.0	1400000	6.0
...	...	...	...	...	...	...
205324	2016.0	FullStack Engineer	wos xzntqzvnxgzvr	2021.0	1500000	7.0
205326	2019.0	FullStack Engineer	xzegojo	2021.0	1200000	4.0
205327	2015.0	Data Scientist	wgbuzgcv wgznqvwn	2021.0	1000000	8.0
205328	2019.0	Data Scientist	ahzzyhbmj	2021.0	1100000	4.0
205329	2017.0	Frontend Engineer	ertdnqvat ojontbo rbn	2021.0	1100000	6.0

153178 rows × 6 columns

Standardization & Encoding

In [26]:

```
X = data

y = data['job_position']
```

In [27]:
```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

X['job_position'] = le.fit_transform(X['job_position'])

y = le.transform(y)
```

```
/var/folders/2m/svsbyfss4h53t29lk1vcnvf40000gn/T/ipykernel_7376/199861872.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#r
eturning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returnin
g-a-view-versus-a-copy)
  X['job_position'] = le.fit_transform(X['job_position'])
```

In [28]:
```
X = X
y = X['company_hash']
```

In [29]:
```
from sklearn.preprocessing import LabelEncoder


le = LabelEncoder()

X['company_hash'] = le.fit_transform(X['company_hash'])

y = le.transform(y)
```

```
/var/folders/2m/svsbyfss4h53t29lk1vcnvf40000gn/T/ipykernel_7376/1388851734.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#r
eturning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returnin
g-a-view-versus-a-copy)
  X['company_hash'] = le.fit_transform(X['company_hash'])
```

In [30]:
```
data=X
```

In [31]:
```
data
```

Out[31]:

	orgyear	job_position	company_hash	ctc_updated_year	ctc	years_of_Experience
0	2016.0	458	870	2020.0	1100000	7.0
1	2018.0	292	18095	2019.0	449999	5.0
2	2015.0	140	14223	2020.0	2000000	8.0
3	2017.0	140	11066	2019.0	700000	6.0
4	2017.0	292	18560	2019.0	1400000	6.0
...	...	...	...	...	...	...
205324	2016.0	292	28211	2021.0	1500000	7.0
205326	2019.0	292	31034	2021.0	1200000	4.0
205327	2015.0	208	27490	2021.0	1000000	8.0
205328	2019.0	208	489	2021.0	1100000	4.0
205329	2017.0	287	5388	2021.0	1100000	6.0

153178 rows × 6 columns

In [32]:
```
cols = X.columns
```

In [33]:
```
from sklearn.preprocessing import MinMaxScaler

ms = MinMaxScaler()

X = ms.fit_transform(X)
```

In [34]:
```
X = pd.DataFrame(X, columns=[cols])
```

In [35]:
```
X.head()
```

Out[35]:

	orgyear	job_position	company_hash	ctc_updated_year	ctc	years_of_Experience
0	0.867925	0.450787	0.025444	0.833333	0.00550	0.132075
1	0.905660	0.287402	0.529202	0.666667	0.00225	0.094340
2	0.849057	0.137795	0.415962	0.833333	0.01000	0.150943
3	0.886792	0.137795	0.323633	0.666667	0.00350	0.113208
4	0.886792	0.287402	0.542801	0.666667	0.00700	0.113208

Manual Clustering

In [36]:
```
sf_company_hash = data.groupby('company_hash').agg({'ctc' : ['mean', 'median', 'max', 'min', 'count']}).reset_i
```

In [37]:

```
sf_company_hash
```

Out[37]:

	company_hash	ctc				
		mean	median	max	min	count
0	0	100000.0	100000.0	100000	100000	1
1	1	300000.0	300000.0	300000	300000	1
2	2	550000.0	550000.0	830000	270000	2
3	3	1100000.0	1100000.0	1100000	1100000	1
4	4	250000.0	250000.0	250000	250000	1
...	...	...	...	...	...	...
34189	34189	2400000.0	2400000.0	2400000	2400000	1
34190	34190	940000.0	940000.0	940000	940000	1
34191	34191	1370000.0	1370000.0	1370000	1370000	1
34192	34192	600000.0	600000.0	600000	600000	1
34193	34193	720000.0	720000.0	720000	720000	1

34194 rows × 6 columns

In [38]:

```
sf_job_position = data.groupby('job_position').agg({'ctc' : ['mean', 'median', 'max', 'min', 'count']}).reset_i
```

In [39]:
```
sf_job_position
```

Out[39]:

	job_position	ctc				
		mean	median	max	min	count
0	0	1200000.0	1200000.0	1200000	1200000	1
1	1	700000.0	700000.0	700000	700000	1
2	2	600000.0	600000.0	600000	600000	1
3	3	470000.0	470000.0	470000	470000	1
4	4	420000.0	420000.0	420000	420000	1
...	...	...	...	...	...	...
1012	1012	1715000.0	1715000.0	2400000	1030000	2
1013	1013	2000000.0	2000000.0	2000000	2000000	1
1014	1014	500000.0	500000.0	500000	500000	1
1015	1015	610000.0	610000.0	610000	610000	1
1016	1016	82000.0	82000.0	82000	82000	1

1017 rows × 6 columns

In [40]:
```
_Experience = data.groupby('years_of_Experience').agg({'ctc' : ['mean', 'median', 'max', 'min', 'count']}).reset
```

```
In [41]: sf_years_of_Experience
```

Out[41]:

	years_of_Experience	ctc				
		mean	median	max	min	count
0	0.00000	1.429270e+07	740000.0	200000000	10000	179
1	1.00000	7.209064e+06	800000.0	200000000	2000	605
2	2.00000	4.247075e+06	650000.0	200000000	3400	2377
3	3.00000	2.268745e+06	670000.0	200000000	24	6955
4	4.00000	1.633791e+06	700000.0	200000000	1000	13868
5	5.00000	1.815167e+06	750000.0	200000000	500	19004
6	6.00000	2.083714e+06	800000.0	200000000	1000	17928
7	7.00000	2.332430e+06	869999.0	200000000	25	17839
8	8.00000	2.432603e+06	950000.0	200000000	1000	16179
9	8.11725	2.732952e+06	905000.0	100000000	8000	62

In [42]:
```
data
```

Out[42]:

	orgyear	job_position	company_hash	ctc_updated_year	ctc	years_of_Experience
0	2016.0	458	870	2020.0	1100000	7.0
1	2018.0	292	18095	2019.0	449999	5.0
2	2015.0	140	14223	2020.0	2000000	8.0
3	2017.0	140	11066	2019.0	700000	6.0
4	2017.0	292	18560	2019.0	1400000	6.0
...	...	...	...	...	...	...
205324	2016.0	292	28211	2021.0	1500000	7.0
205326	2019.0	292	31034	2021.0	1200000	4.0
205327	2015.0	208	27490	2021.0	1000000	8.0
205328	2019.0	208	489	2021.0	1100000	4.0
205329	2017.0	287	5388	2021.0	1100000	6.0

153178 rows × 6 columns

Unsupervised learning

In [43]:
```
data=pd.DataFrame(data, columns=['orgyear','job_position','company_hash','ctc','ctc_updated_year', 'years_of_Exp
```

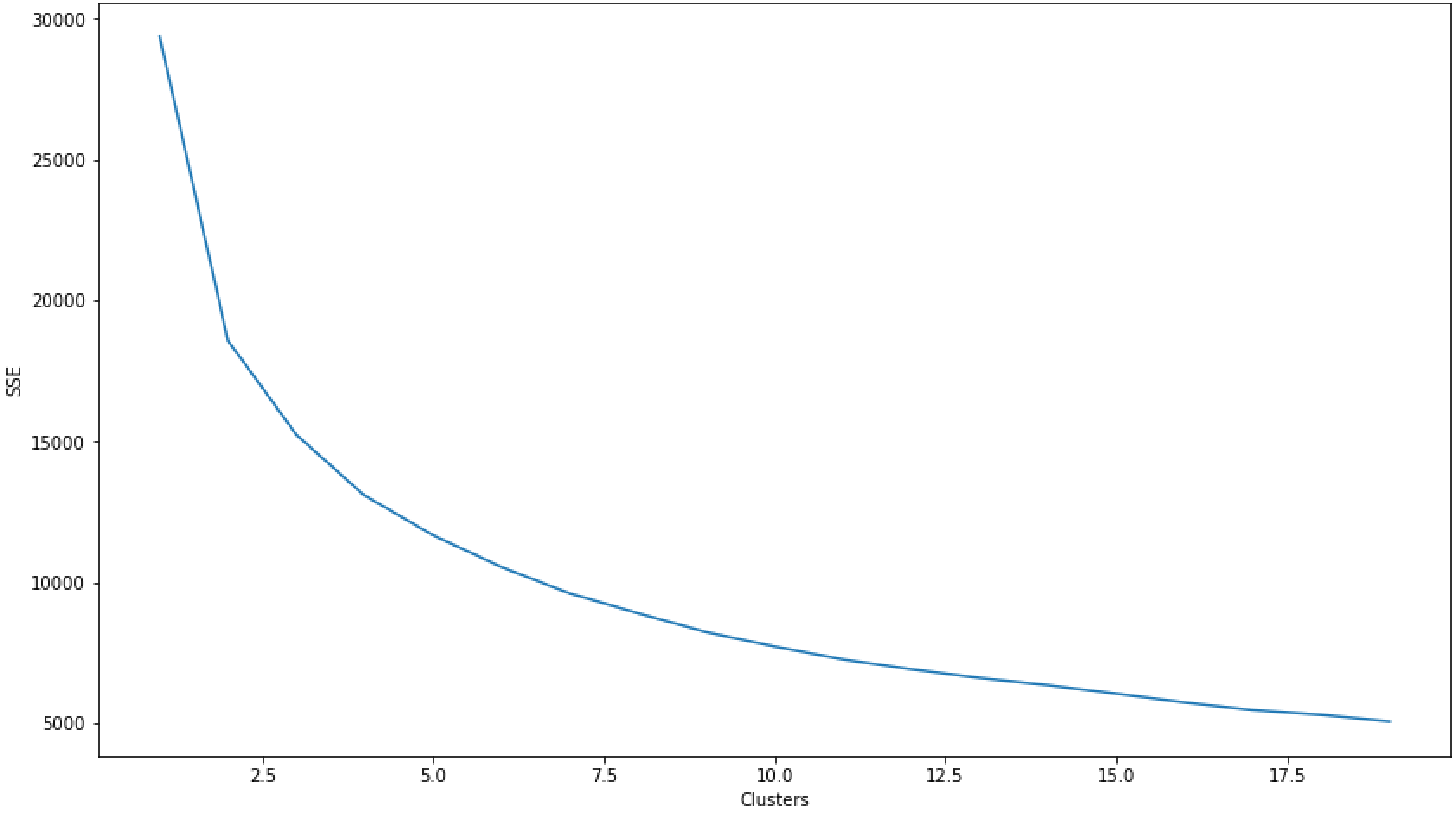
K- means clustering

In [44]:

```
from sklearn.cluster import KMeans

Inter = []
for i in range(1,20):
    model = KMeans(n_clusters = i)
    model.fit(X)
    Inter.append(model.inertia_)

# plotting the Elbow
plt.figure(figsize = (14, 8))
plt.plot(np.arange(1,20), Inter)
plt.xlabel('Clusters')
plt.ylabel('SSE')
plt.show()
```



Hierarchical Clustering

In [45]:

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 6)
X_principal = pca.fit_transform(X)
X_principal = pd.DataFrame(X_principal)
X_principal.columns = ['P1', 'P2','P3','P4','P5','P6']

X_principal.head(2)
```

Out[45]:

```
          P1         P2         P3         P4         P5            P6
0   0.480169  -0.151548   0.142753  -0.022936  -0.007883  9.909772e-14
1  -0.003887   0.071486  -0.003421  -0.098304  -0.006352 -6.177225e-16
```

In [ ]:

```
import scipy.cluster.hierarchy as shc
plt.figure(figsize =(6, 6))
plt.title('Visualising the data')
Dendrogram = shc.dendrogram((shc.linkage(X_principal, method ='ward')))
```

Actionable Insights & Recommendations

In [ ]:

In [ ]:

In [ ]: