In [1]:
```
#importing required packages
import pandas as pd
import numpy as np
import re
import matplotlib.pyplot as plt
from datetime import datetime
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.impute import KNNImputer, SimpleImputer
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.metrics import silhouette_score
from scipy.spatial.distance import cdist
from sklearn.neighbors import NearestNeighbors
```

In [2]:
```
#reading and loading the data set
df = pd.read_csv('scaler_clustering.csv')
```

In [3]:
```
# Drop the 'Unnamed: 0' column
df.drop(columns=['Unnamed: 0'], inplace=True)
```

In [4]:
```
#shape of the data set
print("Shape of the dataset:", df.shape)
```

```
Shape of the dataset: (205843, 6)
```

In [5]:
```
#features and their data types
print("\nInformation about the dataset:\n", df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205843 entries, 0 to 205842
Data columns (total 6 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   company_hash      205799 non-null  object 
 1   email_hash        205843 non-null  object 
 2   orgyear           205757 non-null  float64
 3   ctc               205843 non-null  int64  
 4   job_position      153279 non-null  object 
 5   ctc_updated_year  205843 non-null  float64
dtypes: float64(2), int64(1), object(3)
memory usage: 9.4+ MB

Information about the dataset:
 None
```

In [6]:
```
#statistical summary
print("\nStatistical summary of numerical attributes:\n", df.describe())
```

```
Statistical summary of numerical attributes:
              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 [7]:
```
#missing values
print("\n missing values:\n", df.isnull().sum())
```

```
 missing values:
 company_hash           44
email_hash               0
orgyear                 86
ctc                      0
job_position         52564
ctc_updated_year         0
dtype: int64
```

In [8]:
```
# Check unique emails and frequency
email_counts = df['email_hash'].value_counts()
print("Total emails:", email_counts.shape[0])
print("Unique Email counts : " ,  email_counts[email_counts == 1].shape[0])
print("Count of Emails with multiple records:\n",  email_counts[email_counts > 1
print("Emails with multiple records:\n",  email_counts[email_counts > 1])
```

```
Total emails: 153443
Unique Email counts :  112227
Count of Emails with multiple records:
 (41216,)
Emails with multiple records:
 email_hash
bbace3cc586400bbc65765bc6a16b77d8913836cfc98b77c05488f02f5714a4b    10
3e5e49daa5527a6d5a33599b238bf9bf31e85b9efa9a94f1c88c5e15a6f31378     9
298528ce3160cc761e4dc37a07337ee2e0589df251d73645aae209b010210eee     9
6842660273f70e9aa239026ba33bfe82275d6ab0d20124021b952b5bc3d07e6c     9
d598d6f1fb21b45593c2afc1c2f76ae9f4cb7167156cdf93246d4192a89d8065     8
                                                                    ..
2001ec1f394b0e783e8368ebda4f913e98b4bf876a307d08c6ab9c90d6cf0069     2
e0580056b68a2566c4714afc0a0c4f04eec881fbb49bb62e542101dc7647315e     2
5e03a50d13d475e1ebdf82008abd5e1dc06a62f6e8df25c0fac4659fa67cad52     2
6b9d65aae5c59e401294f7c652e20f29c74e47d76877720736ae492b01774a08     2
b1e44894a7d09a75652cfa26c641e206f7fa8c58c7c1a10eca269b350404daef     2
Name: count, Length: 41216, dtype: int64
```

In [9]:
```
# Convert categorical columns to 'category' dtype
categorical_cols = ['email_hash', 'company_hash', 'job_position']
for col in categorical_cols:
    df[col] = df[col].astype('category')

print("\nUpdated data types:\n", df.dtypes)
```

```
Updated data types:
 company_hash        category
email_hash           category
orgyear               float64
ctc                     int64
job_position         category
ctc_updated_year      float64
dtype: object
```

In [10]:
```
# Data Cleaning

# Remove special characters from Company_hash and Job_position using regex
df['company_hash'] = df['company_hash'].astype(str).apply(lambda x: re.sub('[^A-
df['job_position'] = df['job_position'].astype(str).apply(lambda x: re.sub('[^A-
```

In [11]:
```
# Check and drop duplicates
print("Duplicates:", df.duplicated().sum())
df = df.drop_duplicates()
```

```
Duplicates: 34
```

In [12]:
```
# updating the nan column with new string Missing since nan is the most frequene
df['job_position'] = df['job_position'].replace('nan','Missing')
```

```
<ipython-input-12-9a430a3ec8ac>:2: 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/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['job_position'] = df['job_position'].replace('nan','Missing')
```

In [13]:
```
# --- Missing Value Imputation ---Impute missing values (if any) - For numeric c
num_cols = ['orgyear']
imputer = KNNImputer(n_neighbors=5)
df[num_cols] = imputer.fit_transform(df[num_cols])

cat_missing = ['job_position','company_hash']

freq_imputer = SimpleImputer(strategy = 'most_frequent') # mode
for col in cat_missing:
    df[col] = pd.DataFrame(freq_imputer.fit_transform(pd.DataFrame(df[col])))
```

In [14]:
```
#orgyear column has year with values less / more than 3 digits. so replacing tho
df['orgyear'] = df['orgyear'].astype(int)
df['orgyear'] = df['orgyear'].apply(
    lambda x: df['orgyear'].mode()[0] if len(str(x)) <= 3 else x
)
df['orgyear'] = df['orgyear'].apply(
    lambda x: df['orgyear'].mode()[0] if len(str(x)) >= 5 else x
)
```

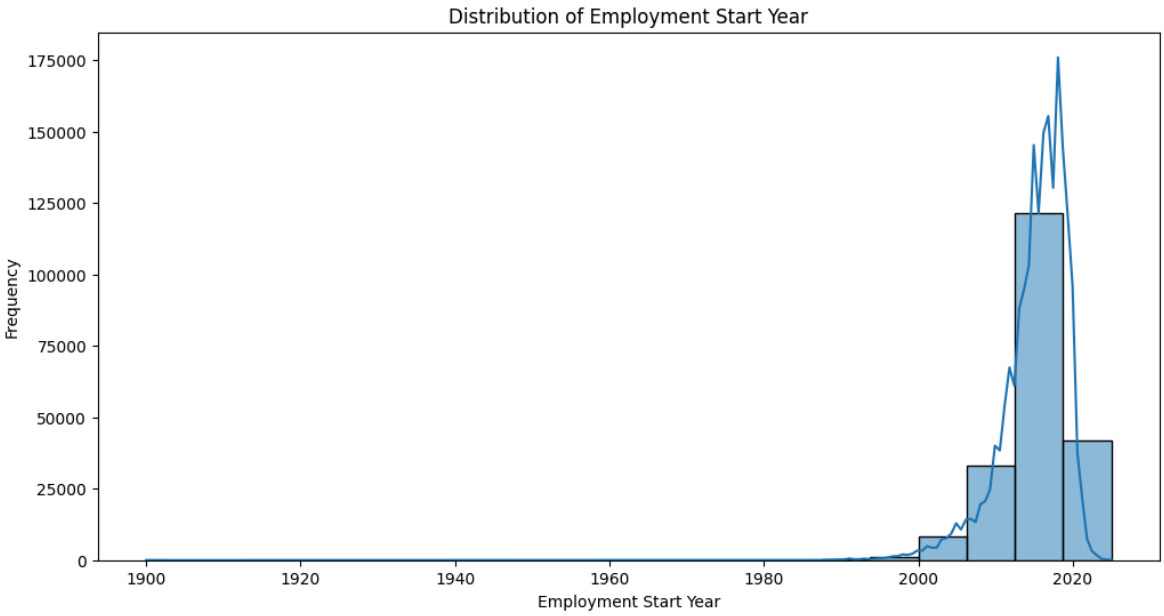
In [15]:
```
# Since  the data set has orgyear value more than 2025 - current year, updating
def correct_orgyear(year):
    if year > 2025:
        return 2000 + year % 10 # assuming that those values are years in 2000s
    return year

# Apply the correction function to the 'orgyear' column
df['orgyear'] = df['orgyear'].apply(correct_orgyear)
```

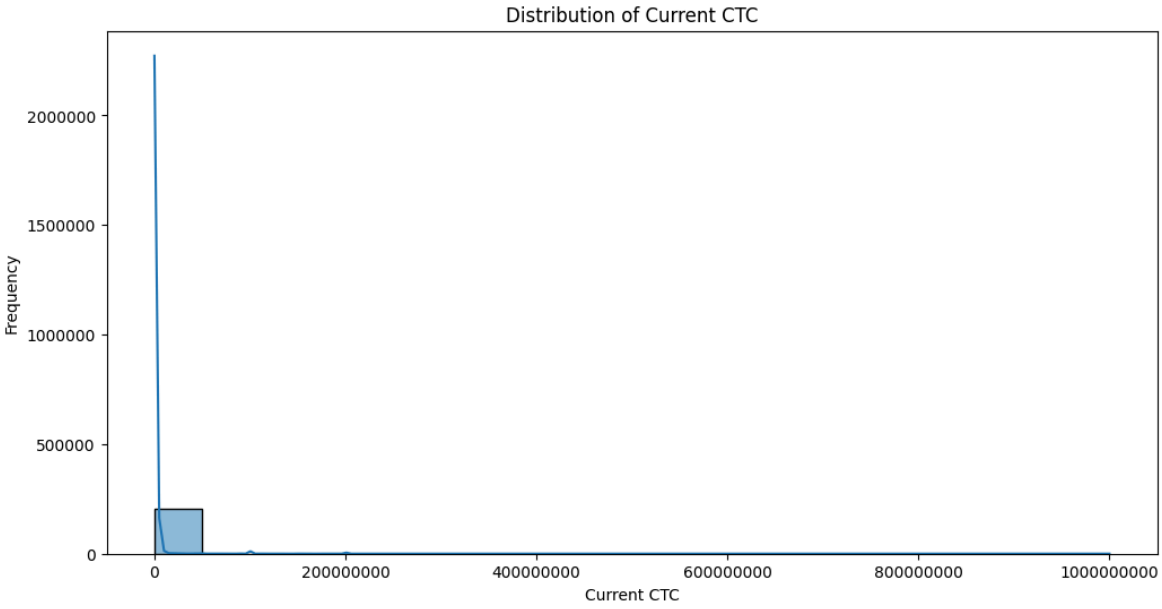
In [16]:
```
# data type change from float to int - ctc_updated_year
df['ctc_updated_year'] = df['ctc_updated_year'].astype(int)
df['ctc_updated_year'].unique()
```

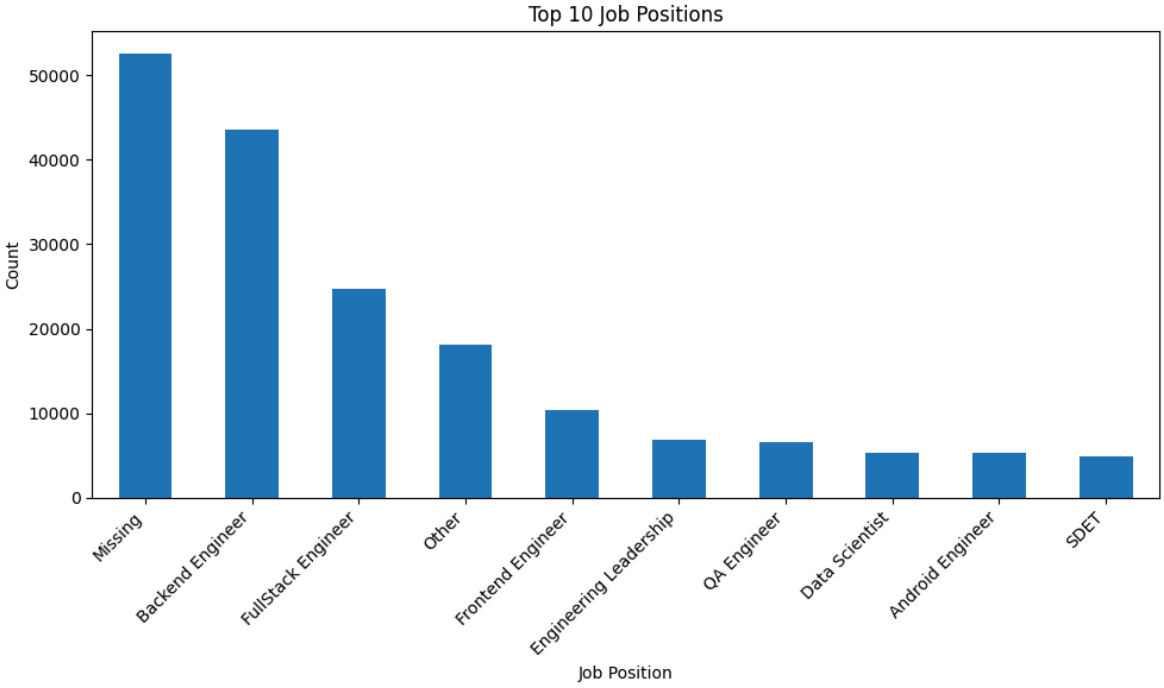
Out[16]:
```
array([2020, 2019, 2021, 2017, 2016, 2015, 2018])
```

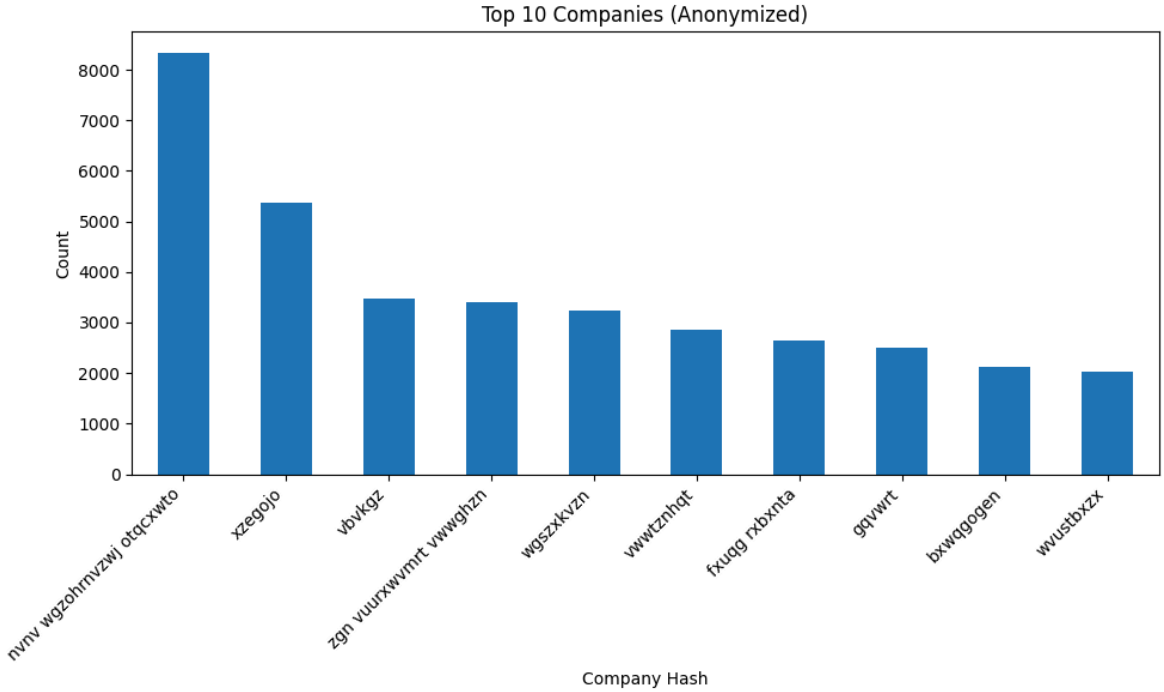
In [17]:
```
# --- Creating 'Years of Experience' Feature ---
current_year =  datetime.now().year # As per the problem context

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

In [18]:
```
#Univariate analysis
plt.figure(figsize=(12, 6))
sns.histplot(df['orgyear'].dropna(), bins=20, kde=True)
plt.title('Distribution of Employment Start Year')
plt.xlabel('Employment Start Year')
plt.ylabel('Frequency')
plt.show()

plt.figure(figsize=(12, 6))
plt.ticklabel_format(style='plain')
sns.histplot(df['ctc'].dropna(), bins=20, kde=True)
plt.title('Distribution of Current CTC')
plt.xlabel('Current CTC')
plt.ylabel('Frequency')
plt.show()

plt.figure(figsize=(10, 6))
df['job_position'].value_counts().nlargest(10).plot(kind='bar')
plt.title('Top 10 Job Positions')
plt.xlabel('Job Position')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

plt.figure(figsize=(10, 6))
df['company_hash'].value_counts().nlargest(10).plot(kind='bar')
plt.title('Top 10 Companies (Anonymized)')
plt.xlabel('Company Hash')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

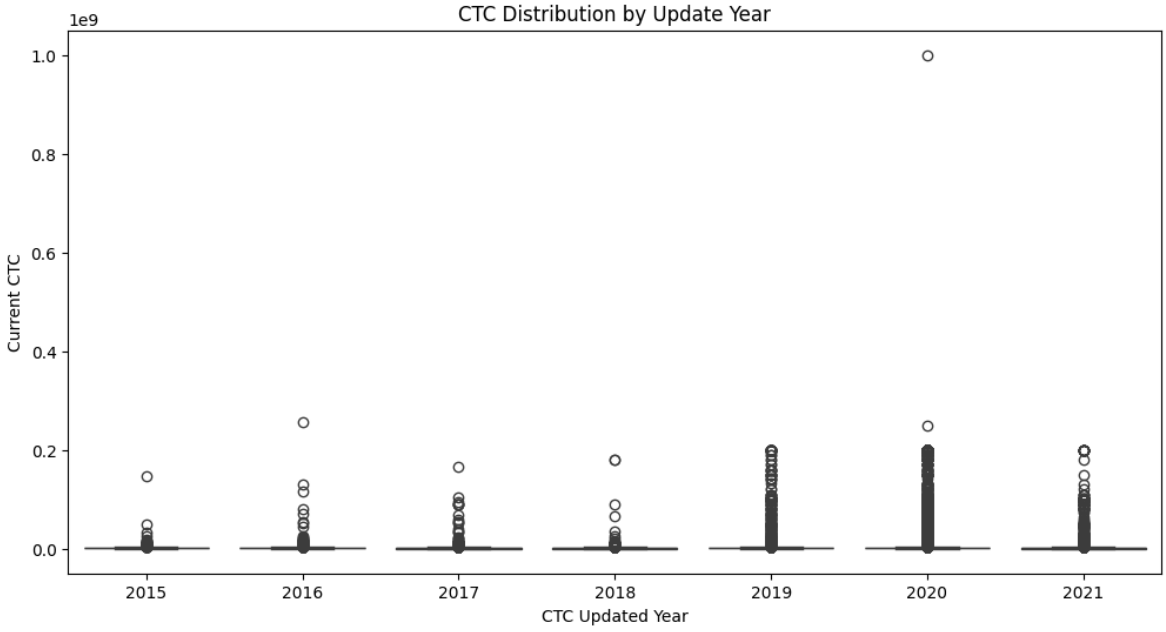


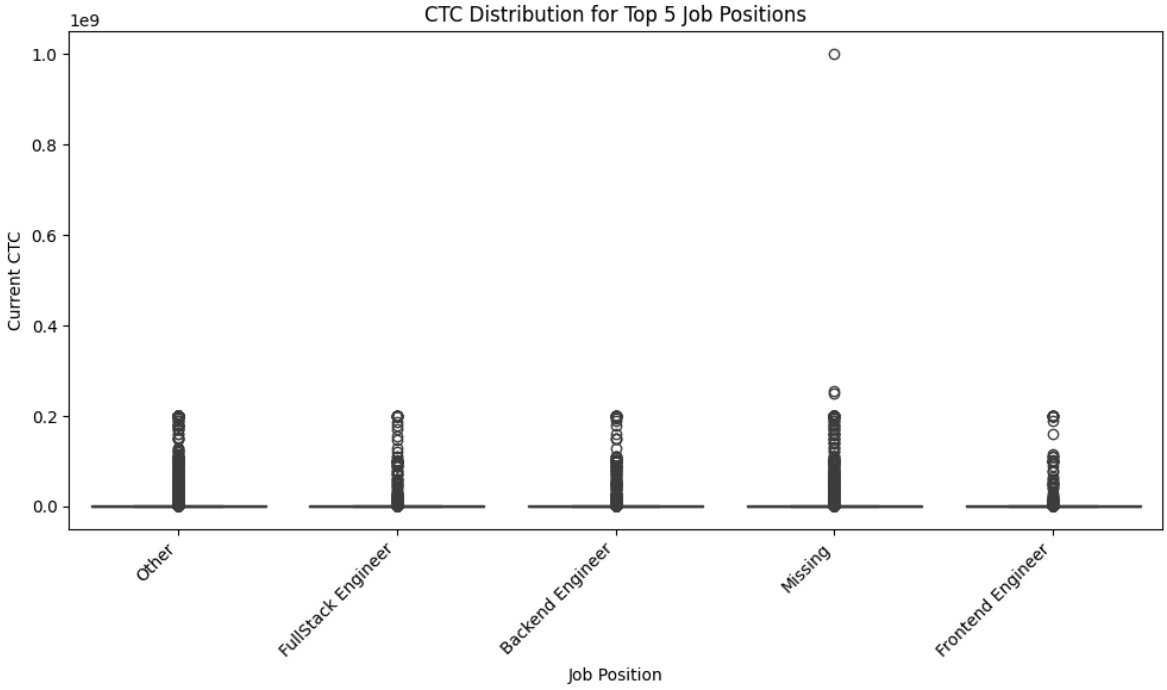

Distribution of Current CTC
Frequency
2000000
1500000
1000000
500000
0
0
200000000
400000000
600000000
800000000
1000000000
Current CTC

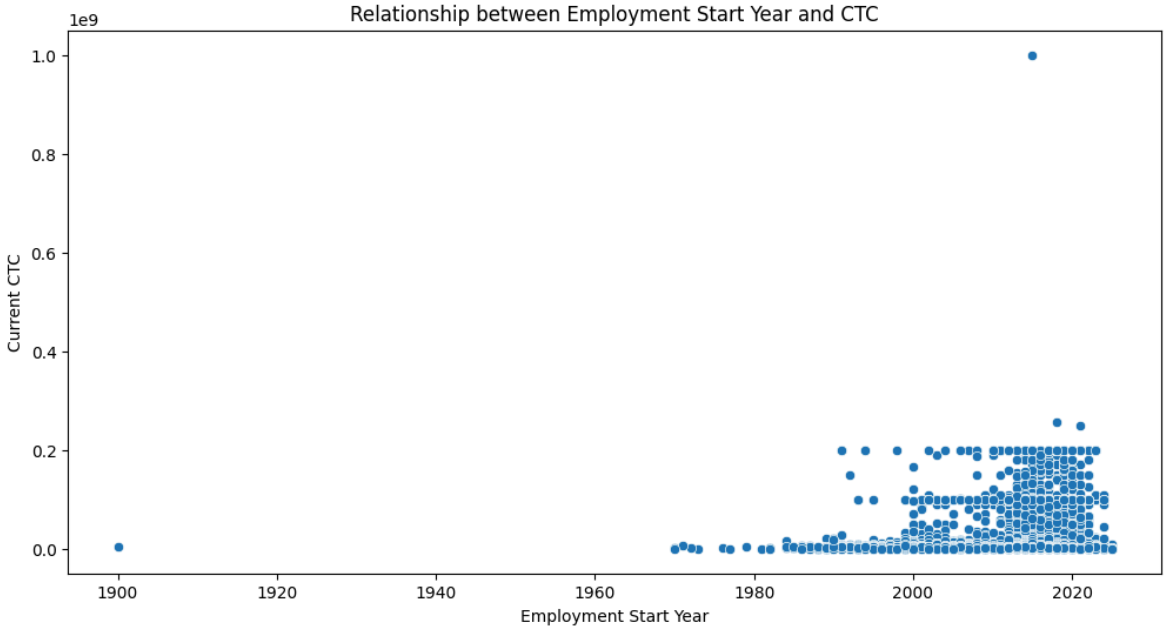

Top 10 Job Positions
Count
50000
40000
30000
20000
10000
0
Missing
Backend Engineer
FullStack Engineer
Other
Frontend Engineer
Engineering Leadership
QA Engineer
Data Scientist
Android Engineer
SDET
Job Position


Top 10 Companies (Anonymized)
Count
8000
7000
6000
5000
4000
3000
2000
1000
0
nvnv wgzohrnvzwj otqcxwto
xzegojo
vbvkgz
zgn vuurxwvmrt vwwghzn
wgszxkvzn
vwwtznhqt
fxuqg rxbxnta
gqvwrt
bxwqgogen
wvustbxzx
Company Hash

In [19]:
```
# --- Bivariate Analysis ---
plt.figure(figsize=(12, 6))
sns.boxplot(x='ctc_updated_year', y='ctc', data=df.dropna())
plt.title('CTC Distribution by Update Year')
plt.xlabel('CTC Updated Year')
plt.ylabel('Current CTC')
plt.show()

# Since 'Company_hash' has many unique values, let's look at the relationship be
top_jobs = df['job_position'].value_counts().nlargest(5).index
subset_df = df[df['job_position'].isin(top_jobs)].dropna(subset=['ctc'])
plt.figure(figsize=(10, 6))
sns.boxplot(x='job_position', y='ctc', data=subset_df)
plt.title('CTC Distribution for Top 5 Job Positions')
plt.xlabel('Job Position')
plt.ylabel('Current CTC')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
# Comment: Compare the CTC ranges for the most common job positions.

# Similarly, we can explore the relationship between 'orgyear' and 'CTC'.
plt.figure(figsize=(12, 6))
sns.scatterplot(x='orgyear', y='ctc', data=df.dropna())
plt.title('Relationship between Employment Start Year and CTC')
plt.xlabel('Employment Start Year')
plt.ylabel('Current CTC')
plt.show()
```







In [20]:
```
# --- Label Encoding for Categorical Features (for clustering later) ---
label_encoders = {}
categorical_cols_for_encoding = ['company_hash', 'job_position']
for col in categorical_cols_for_encoding:
    le = LabelEncoder()
    df[col + '_encoded'] = le.fit_transform(df[col])
    label_encoders[col] = le

print("\nSample of encoded categorical features:\n", df[['company_hash', 'compan

# --- Standardization for Numerical Features (for clustering later) ---
numerical_cols_for_scaling = ['ctc', 'years_of_experience']
scaler = StandardScaler()
for col in numerical_cols_for_scaling:
    # Reshape the column using values.reshape(-1, 1)
    df[col + '_scaled'] = scaler.fit_transform(df[[col]])
print("\nSample of scaled numerical features:\n", df[['ctc', 'ctc_scaled', 'year
```

```
Sample of encoded categorical features:
                company_hash  company_hash_encoded        job_position  \
0            atrgxnnt xzaxv                   969               Other
1  qtrxvzwt xzegwgbb rxbxnta                 19730  FullStack Engineer
2             ojzwnvwnxw vx                 15512    Backend Engineer
3                 ngpgutaxv                 12108    Backend Engineer
4                qxen sqghu                 20226  FullStack Engineer

   job_position_encoded
0                   455
1                   289
2                   138
3                   138
4                   289

Sample of scaled numerical features:
       ctc  ctc_scaled  years_of_experience  years_of_experience_scaled
0  1100000   -0.099295                    9                   -0.208606
1   449999   -0.154371                    7                   -0.680044
2  2000000   -0.023036                   10                    0.027113
3   700000   -0.133188                    8                   -0.444325
4  1400000   -0.073875                    8                   -0.444325
```

In [21]:

```python
# --- Manual Clustering and CTC Analysis ---
def analyze_ctc_group(group):
    return group['ctc'].agg(['mean', 'median', 'max', 'min', 'count'])

# --- 1. Company, Job Position, Years of Experience Level ---
grouped_co_job_exp = df.groupby(['company_hash', 'job_position', 'years_of_exper
ctc_summary_co_job_exp = grouped_co_job_exp.apply(analyze_ctc_group).reset_index
ctc_summary_co_job_exp.rename(columns={'mean': 'avg_ctc_co_job_exp', 'median': '

df_merged_co_job_exp = pd.merge(df, ctc_summary_co_job_exp, on=['company_hash',

def designation_flag(row):
    if row['ctc'] > row['avg_ctc_co_job_exp']:
        return 1
    elif row['ctc'] < row['median_ctc_co_job_exp']:
        return 3
    else:
        return 2

df_merged_co_job_exp['Designation'] = df_merged_co_job_exp.apply(designation_fla
print("\nSample with Designation Flag:\n", df_merged_co_job_exp[['company_hash',

print("\nDesignation Flag Value Counts:\n", df_merged_co_job_exp['Designation'].
```

```
<ipython-input-21-a97aec29f3ed>:7: DeprecationWarning: DataFrameGroupBy.apply ope
rated on the grouping columns. This behavior is deprecated, and in a future versi
on of pandas the grouping columns will be excluded from the operation. Either pas
s `include_groups=False` to exclude the groupings or explicitly select the groupi
ng columns after groupby to silence this warning.
  ctc_summary_co_job_exp = grouped_co_job_exp.apply(analyze_ctc_group).reset_inde
x()
```

```
Sample with Designation Flag:
                company_hash        job_position  years_of_experience  \
0             atrgxnnt xzaxv               Other                    9
1  qtrxvzwt xzegwgbb rxbxnta  FullStack Engineer                    7
2              ojzwnvwnxw vx    Backend Engineer                   10
3                  ngpgutaxv    Backend Engineer                    8
4                qxen sqghu  FullStack Engineer                    8

       ctc  avg_ctc_co_job_exp  median_ctc_co_job_exp  Designation
0  1100000        1.100000e+06              1100000.0            2
1   449999        7.742856e+05               750000.0            3
2  2000000        2.000000e+06              2000000.0            2
3   700000        1.037500e+06               950000.0            3
4  1400000        1.400000e+06              1400000.0            2

Designation Flag Value Counts:
 Designation
2    119531
3     47716
1     38562
Name: count, dtype: int64
```

In [22]:

```python
# --- 2. Company & Job Position Level ---
grouped_co_job = df.groupby(['company_hash', 'job_position'])
ctc_summary_co_job = grouped_co_job.apply(analyze_ctc_group).reset_index()
ctc_summary_co_job.rename(columns={'mean': 'avg_ctc_co_job', 'median': 'median_c

df_merged_co_job = pd.merge(df_merged_co_job_exp, ctc_summary_co_job, on=['compa

def class_flag(row):
    if row['ctc'] > row['avg_ctc_co_job']:
        return 1
    elif row['ctc'] < row['median_ctc_co_job']:
        return 3
    else:
        return 2

df_merged_co_job['Class'] = df_merged_co_job.apply(class_flag, axis=1)
print("\nSample with Class Flag:\n", df_merged_co_job[['company_hash', 'job_posi

print("\nClass Flag Value Counts:\n", df_merged_co_job['Class'].value_counts())
```

```
<ipython-input-22-08398003fe5e>:3: DeprecationWarning: DataFrameGroupBy.apply ope
rated on the grouping columns. This behavior is deprecated, and in a future versi
on of pandas the grouping columns will be excluded from the operation. Either pas
s `include_groups=False` to exclude the groupings or explicitly select the groupi
ng columns after groupby to silence this warning.
  ctc_summary_co_job = grouped_co_job.apply(analyze_ctc_group).reset_index()
```

```
Sample with Class Flag:
                company_hash        job_position      ctc  avg_ctc_co_job  \
0             atrgxnnt xzaxv               Other  1100000    1.085000e+06
1  qtrxvzwt xzegwgbb rxbxnta  FullStack Engineer   449999    5.514409e+06
2              ojzwnvwnxw vx    Backend Engineer  2000000    2.000000e+06
3                  ngpgutaxv    Backend Engineer   700000    1.159240e+06
4                  qxen sqghu  FullStack Engineer  1400000    1.054000e+06

   median_ctc_co_job  Class
0          1085000.0      1
1           875000.0      3
2          2000000.0      2
3          1050000.0      3
4          1400000.0      1

Class Flag Value Counts:
 Class
2    88515
3    70590
1    46704
Name: count, dtype: int64
```

In [23]:

```python
# --- 3. Company Level ---
grouped_co = df.groupby(['company_hash'])
ctc_summary_co = grouped_co.apply(analyze_ctc_group).reset_index()
ctc_summary_co.rename(columns={'mean': 'avg_ctc_co', 'median': 'median_ctc_co'},

df_merged_co = pd.merge(df_merged_co_job, ctc_summary_co, on=['company_hash'], h

def tier_flag(row):
    if row['ctc'] > row['avg_ctc_co']:
        return 1
    elif row['ctc'] < row['median_ctc_co']:
        return 3
    else:
        return 2

df_merged_co['Tier'] = df_merged_co.apply(tier_flag, axis=1)
print("\nSample with Tier Flag:\n", df_merged_co[['company_hash', 'ctc', 'avg_ct

print("\nTier Flag Value Counts:\n", df_merged_co['Tier'].value_counts())
```

```
<ipython-input-23-5bd9789d32fa>:3: DeprecationWarning: DataFrameGroupBy.apply ope
rated on the grouping columns. This behavior is deprecated, and in a future versi
on of pandas the grouping columns will be excluded from the operation. Either pas
s `include_groups=False` to exclude the groupings or explicitly select the groupi
ng columns after groupby to silence this warning.
  ctc_summary_co = grouped_co.apply(analyze_ctc_group).reset_index()
```

```
Sample with Tier Flag:
                company_hash      ctc    avg_ctc_co  median_ctc_co  Tier
0             atrgxnnt xzaxv  1100000  1.087778e+06      1070000.0     1
1  qtrxvzwt xzegwgbb rxbxnta   449999  2.296193e+06       900000.0     3
2              ojzwnvwnxw vx  2000000  2.000000e+06      2000000.0     2
3                  ngpgutaxv   700000  1.452014e+06      1100000.0     3
4                  qxen sqghu  1400000  1.418667e+06      1550000.0     3

Tier Flag Value Counts:
 Tier
3    83424
2    75232
1    47153
Name: count, dtype: int64
```

In [24]:
```
# Top 10 employees earning more than most in company (Tier 1)
top_tier1 = df_merged_co[df_merged_co['Tier'] == 1].sort_values('ctc', ascending
print("Top 10 employees (Tier 1):")
print(top_tier1[['email_hash', 'company_hash', 'job_position', 'ctc', 'Tier']])
```

```
Top 10 employees (Tier 1):
                                              email_hash  \
117626  5b4bed51797140db4ed52018a979db1e34cee49e27b488...
22387   94970774b1cf64e61cf30fb6541cd27fdb31b220cda54b...
22286   3e7804b3aef9f10977903287530bb816dcde2d98e87bf3...
22185   97d25613e7bc3f47c87492d311f77232c105e4bc9ce642...
36253   1bdd2d3f1509045bd303e67882df623b0f892d0509b6e8...
22089   f4e874b3329098fdb3de47a83e1b41b2f5f4b873e148dd...
19712   59316048d113539202325e05af9b66620255ba84eab635...
126190  0c9c37269bd373ef507df0bc1bb318787fd895c858b74e...
10401   74f506e2567fb54995842894d2021582effbcde027d8e3...
99280   2744c7f42fd4d492fa66cb2ba5168921c444dc8611ffa2...

                   company_hash            job_position        ctc  Tier
117626              wxowg ojontbo                 Missing  255555555     1
22387                          zv  Engineering Leadership  200000000     1
22286                exqonoghqwt            Data Analyst  200000000     1
22185   nvnv wgzohrnvzwj otqcxwto         Support Engineer  200000000     1
36253            ywr ntwyzgrgsxto         Android Engineer  200000000     1
22089               xqgz bghznvxz                    Other  200000000     1
19712   nvnv wgzohrnvzwj otqcxwto            Data Analyst  200000000     1
126190                     sggsrt                 Missing  200000000     1
10401            mtwngz axwpxzogz              QA Engineer  200000000     1
99280                   bxwqgogen         Android Engineer  200000000     1
```

In [37]:
```
# Top 10 data science employees earning more than peers (Class 1)
ds_top_class1 = df_merged_co_job[(df_merged_co_job['job_position'].str.contains(
top_10_ds_class1 = ds_top_class1.groupby('company_hash').apply(lambda x: x.sort_
print("Top 10 Data Science employees per company (Class 1):")
print(top_10_ds_class1[['email_hash', 'company_hash', 'job_position', 'ctc', 'Cl
```

```
Top 10 Data Science employees per company (Class 1):
                                              email_hash  \
1637  268a5aa92f0b6d0c675fc9cc1e300eb0c5930a3a139a23...
689   f5b2a30853a67e1703249db6003884d7e1ae69e0c03aa0...
605   979d02840c45c1d5790306130a0977aab05f2bd2679687...
633   655da5cd99f1ba4ad249dade5039b914023484fb7f3959...
851   35d4845547c5d2e0c2eadc197c97c678035bceb5fddd2d...
1300  9ce2995b2221fe627e861daea9d0603872cce8cc128390...
1252  2f9a4241053f76b2f8c50ea593a90586d38b3f0e08c141...
374   6d4a5d19e889596252b038ee0409510aec8c0b32007fb9...
1290  aad581a532f319c76c6e73937572feed9867d5ee2f1093...
350   89f343bf01094accb8b0b2c799499daf6bf881321db2e4...

                   company_hash    job_position        ctc  Class
1637                       zgzt  Data Scientist  200000000      1
689                ogwxn szqvrt    Data Analyst  200000000      1
605             ntrtutqegqbvzwt    Data Analyst  200000000      1
633   nvnv wgzohrnvzwj otqcxwto    Data Analyst  200000000      1
851                         qmo    Data Analyst  200000000      1
1300                  wgzahtzn    Data Analyst  200000000      1
1252                  vwwtznhqt    Data Analyst  200000000      1
374                       gnytq    Data Analyst  200000000      1
1290                  wgszxkvzn    Data Analyst  200000000      1
350                fxuqg rxbxnta    Data Analyst  200000000      1
```

```
<ipython-input-37-5bcbee76dd1d>:5: DeprecationWarning: DataFrameGroupBy.apply ope
rated on the grouping columns. This behavior is deprecated, and in a future versi
on of pandas the grouping columns will be excluded from the operation. Either pas
s `include_groups=False` to exclude the groupings or explicitly select the groupi
ng columns after groupby to silence this warning.
  top_10_ds_class1 = ds_top_class1.groupby('company_hash').apply(lambda x: x.sort
_values(by = 'ctc', ascending=False).head(10)).reset_index(drop=True).sort_values
('ctc', ascending=False).head(10)
```

In [39]:

```
# Bottom 10 data science employees earning less than peers (Class 3)
ds_bottom_class3 = df_merged_co_job[(df_merged_co_job['job_position'].str.contai
bottom_10_ds_class3 = ds_bottom_class3.groupby('company_hash').apply(lambda x: x
print("Bottom 10 Data Science employees per company (Class 3):")
print(bottom_10_ds_class3[['email_hash', 'company_hash', 'job_position', 'ctc',
```

```
Bottom 10 Data Science employees per company (Class 3):
                                              email_hash  \
1102  c5b586cc2d3b9e783e76763f274c6fbb05e7fabb12fbcc...
324   ab2dc9db23c3104f0b6b3dbd4cdd5bfb9e5829b8b7943d...
716   bd9c04a574090e05b366a81cdb2f3f565d0c60fa8b1647...
1666  aeb32d3e07a73c021c6ad75a3eebef4bedc726109d853b...
915   13fca3a2e659a7641ac165c4e649947398233b309c6495...
189   690f6fdab1ab7514a6a9325ebd6cfe910dbf12d46b6fde...
1568  648975bbd733a9949d715ba66d2712d0c01ace6e046c9a...
974   8001bc017fbe95541d23f5780c3edb988b7d9b2225e39e...
487   4af25f1052f845426450ad6d96e78338c3b913e3cd4539...
991   4c029c8afc9c245b4300d08f2cc0ccde425aa1a620debe...

                           company_hash    job_position   ctc  Class
1102                    uhmrxwxo ovuxtzn    Data Analyst  7500      3
324           exznqhon ogrhnxgzo ucn rna  Data Scientist  7200      3
716                              onhatzn  Data Scientist  6000      3
1666                          zthonvq xzw    Data Analyst  5000      3
915                 rvntzncxtf vzvrjnxwo    Data Analyst  5000      3
189                  bxyhu wgbbhzxwvnxgz  Data Scientist  4000      3
1568                        yn btaxv rna  Data Scientist  4000      3
974   srgmvrtast  xzntrrxstzwt ge nyxzso  Data Scientist  4000      3
487                                 ihxpq  Data Scientist  4000      3
991                               stzj rvmo  Data Scientist  3500      3
```

```
<ipython-input-39-9bf7cf937ce4>:3: DeprecationWarning: DataFrameGroupBy.apply ope
rated on the grouping columns. This behavior is deprecated, and in a future versi
on of pandas the grouping columns will be excluded from the operation. Either pas
s `include_groups=False` to exclude the groupings or explicitly select the groupi
ng columns after groupby to silence this warning.
  bottom_10_ds_class3 = ds_bottom_class3.groupby('company_hash').apply(lambda x:
x.sort_values(by='ctc').tail(10)).reset_index(drop=True).sort_values('ctc', ascen
ding=False).tail(10)
```

In [27]:
```
# Bottom 10 employees earning less than most in company (Tier 3)
bottom_tier3 = df_merged_co[df_merged_co['Tier'] == 3].sort_values('ctc').head(1
print("Bottom 10 employees (Tier 3):")
print(bottom_tier3[['email_hash', 'company_hash', 'job_position', 'ctc', 'Tier']
```

```
Bottom 10 employees (Tier 3):
                                              email_hash  \
118226  f2b58aeed3c074652de2cfd3c0717a5d21d6fbcf342a78...
114157  23ad96d6b6f1ecf554a52f6e9b61677c7d73d8a409a143...
184918  b8a0bb340583936b5a7923947e9aec21add5ebc50cd60b...
183776  75357254a31f133e2d3870057922feddeba82b88056a07...
116938  f7e5e788676100d7c4146740ada9e2f8974defc01f571d...
166375  c411a6917058b50f44d7c62751be9b232155b23211de4c...
171173  80ba0259f9f59034c4927cf3bd38dc9ce2eb60ff18135b...
150664  9af3dca6c9d705d8d42585ccfce2627f00e1629130d14e...
99417   b995d7a2ae5c6f8497762ce04dc5c04ad6ec734d70802a...
147787  299f764fcae62f331f3c5eb1b451e7107302ded46e2a71...

                        company_hash            job_position   ctc  Tier
118226               zgpxv wgqugqvnxgz                   Other     6     3
114157                       grd sqghu                 Missing    14     3
184918                         buyvoxo                 Missing    15     3
183776                           tbxao                 Missing    16     3
116938  urvj svbto24d7 uqxcvnt rxbxnta                 Missing   200     3
166375                         rxnbho7  Engineering Leadership   300     3
171173                     xzegqbvnxwv        Backend Engineer   600     3
150664                         xzegojo                 Missing   600     3
99417        nvnv wgzohrnvzwj otqcxwto      FullStack Engineer   600     3
147787  tznqvjz tahwvnxgz ntwyzgrgsxto      FullStack Engineer  1000     3
```

In [40]:
```
# Top 10 employees in each company in X department with 5/6/7 years experience e
years_exp_filter = [5, 6, 7]
top_exp_tier = df_merged_co[(df_merged_co['years_of_experience'].isin(years_exp_
top_exp_tier_grouped = top_exp_tier.groupby(['company_hash', 'job_position']).ap
print("Top 10 employees in each company & department with 5/6/7 years experience
print(top_exp_tier_grouped[['email_hash', 'company_hash', 'job_position', 'years
```

```
Top 10 employees in each company & department with 5/6/7 years experience (Tier
1):
                                              email_hash          company_hash  \
6688  5b4bed51797140db4ed52018a979db1e34cee49e27b488...         wxowg ojontbo
4015  431c610cffb5f699476173431bb1f47a51bcc680407e44...          qgmtqn mgowy
5069  48b00207f75dd25ca9d518103e2ddc3c9a9706e51ae393...  uqvbvnx ntwyzgrgsxto
1458  71d7605911c92225343efc7e8aa1a81b60b5ed81796318...        fxuqg rxbxnta
1459  1b95e7ba0ee82100ca5a034239fa0203a1bec14280b82a...        fxuqg rxbxnta
1556  68aa38470922a03f6022280b2a13c6f5ab6a717f70c77a...               gnytq
5520  8dfe6251bd4ec533f02ddceb98b3dcebb9550ccd4ef2e6...              vbvkgz
7924  59361208b0af18838c3240d4f7a02f6aad20ed93f9a73e...                 zvz
6771  431c610cffb5f699476173431bb1f47a51bcc680407e44...        xb  v onhatzn
3413  3c453dd102ae47a4ed1841be352213fad363d0944177e9...        otre tburgjta

           job_position  years_of_experience        ctc  Tier
6688            Missing                    7  255555555     1
4015            Missing                    5  200000000     1
5069            Missing                    5  200000000     1
1458   Frontend Engineer                   7  200000000     1
1459  FullStack Engineer                   7  200000000     1
1556            Missing                    5  200000000     1
5520              Other                    7  200000000     1
7924            Missing                    5  200000000     1
6771  FullStack Engineer                   5  200000000     1
3413    Backend Engineer                   5  200000000     1
```

```
<ipython-input-40-a2931de2f76d>:4: DeprecationWarning: DataFrameGroupBy.apply ope
rated on the grouping columns. This behavior is deprecated, and in a future versi
on of pandas the grouping columns will be excluded from the operation. Either pas
s `include_groups=False` to exclude the groupings or explicitly select the groupi
ng columns after groupby to silence this warning.
  top_exp_tier_grouped = top_exp_tier.groupby(['company_hash', 'job_position']).a
pply(lambda x: x.sort_values(by='ctc',ascending=False).head(10)).reset_index(drop
=True).sort_values('ctc', ascending=False).head(10)
```

In [29]:
```
# Top 10 companies based on average CTC
top_companies = df_merged_co.sort_values('avg_ctc_co', ascending=False).head(10)
print("Top 10 companies based on average CTC:")
print(top_companies)
```

Top 10 companies based on average CTC:

	company_hash	email_hash	orgyear	ctc	job_position	ctc_updated_year	years_of_experience
72824	whmxw rgsxwo uqxcvnt rxbxnta	29a71dd13adf6d2d497571a565bb3096cf66cb46cd1ece...	2015	1000150000	Missing	2020	10
3301	aveegaxr xzntqzvnxgzvr hzxctqoxnj	06d231f167701592a69cdd7d5c825a0f5b30f0347a4078...	2021	250000000	Missing	2020	4
52823	opxrr	57cd2c7b9703fae9ee2e543d48dc8445cff9a36b33349e...	2014	200000000	Missing	2020	11
32673	wqxwwrhmo	c1988a101573e8c3ce2667d33427579285237b7fbfe77f...	2016	200000000	Other	2020	9
2960	xzaxv qvnxzso vza qtotvqwy ucn rna	f958792fd46b8a4453d0c2f95512bcc6aa7e0108bcf047...	2016	200000000	Missing	2020	9
103063	prxtzng	ab8048ef33901acedee6673ad5168672f4d69507984a1e...	2021	200000000	Frontend Engineer	2020	4
106	oxburjyq ogrhnxgzo rru	996aef9bba62bd99d6cb8e8c112c0ec8096b203ae50b97...	2017	200000000	Support Engineer	2020	8
6794	wgbutntzn ojontbo xzw	2323f80afa6f3c809ac468997c1cf1ea8572d06bd8904e...	2017	200000000	Other	2020	8
691	outwnqt vzvrjnxwv	dfdb45fb9631b9064a94be87a27a621068530ac1f3807c...	2017	200000000	Other	2020	8
34562	ofvbx cxctpvzvzav xzonxnhnt ge owxtzwt vza ntw...	a7ff95d399e2822b866066d08467f99711e3894ba79b96...	2017	200000000	Other	2020	8

	company_hash_encoded	job_position_encoded	ctc_scaled	...
72824	30494	422	84.552726	...
3301	1218	422	20.990629	...
52823	16064	422	16.754003	...
32673	31008	455	16.754003	...
2960	33590	422	16.754003	...
103063	18506	285	16.754003	...
106	17542	859	16.754003	...
6794	29925	455	16.754003	...
691	16660	455	16.754003	...
34562	14619	455	16.754003	...

	max_y	min_y	count_y	Class	avg_ctc_co
72824	1.000150e+09	1.000150e+09	1.0	2	1.000150e+09
3301	2.500000e+08	2.500000e+08	1.0	2	2.500000e+08
52823	2.000000e+08	2.000000e+08	1.0	2	2.000000e+08
32673	2.000000e+08	2.000000e+08	1.0	2	2.000000e+08
2960	2.000000e+08	2.000000e+08	1.0	2	2.000000e+08
103063	2.000000e+08	2.000000e+08	1.0	2	2.000000e+08
106	2.000000e+08	2.000000e+08	1.0	2	2.000000e+08
6794	2.000000e+08	2.000000e+08	1.0	2	2.000000e+08
691	2.000000e+08	2.000000e+08	1.0	2	2.000000e+08
34562	2.000000e+08	2.000000e+08	1.0	2	2.000000e+08

```
        median_ctc_co           max           min  count  Tier
72824    1.000150e+09  1.000150e+09  1.000150e+09    1.0     2
3301     2.500000e+08  2.500000e+08  2.500000e+08    1.0     2
52823    2.000000e+08  2.000000e+08  2.000000e+08    1.0     2
32673    2.000000e+08  2.000000e+08  2.000000e+08    1.0     2
2960     2.000000e+08  2.000000e+08  2.000000e+08    1.0     2
103063   2.000000e+08  2.000000e+08  2.000000e+08    1.0     2
106      2.000000e+08  2.000000e+08  2.000000e+08    1.0     2
6794     2.000000e+08  2.000000e+08  2.000000e+08    1.0     2
691      2.000000e+08  2.000000e+08  2.000000e+08    1.0     2
34562    2.000000e+08  2.000000e+08  2.000000e+08    1.0     2

[10 rows x 29 columns]
```

In [42]:
```
# Top 2 positions in every company based on average CTC
pos_ctc = df.groupby(['company_hash', 'job_position'])['ctc'].mean().reset_index
top2_positions = pos_ctc.groupby('company_hash').apply(lambda x: x.sort_values(b
print("Top 2 positions in every company based on average CTC:")
print(top2_positions)
```

```
Top 2 positions in every company based on average CTC:
                       company_hash        job_position        ctc
0                                 0             Missing   100000.0
1                                 0               Other   100000.0
2                              0000               Other  1150000.0
3                        01 ojztqsj   Frontend Engineer   830000.0
4                        01 ojztqsj    Android Engineer   270000.0
...                             ...                 ...        ...
50254                            zz             Missing   500000.0
50255  zzb ztdnstz vacxogqj ucn rna             Missing  3000000.0
50256  zzb ztdnstz vacxogqj ucn rna  FullStack Engineer   600000.0
50257                        zzgato             Missing  1800000.0
50258                        zzzbzb               Other   720000.0

[50259 rows x 3 columns]
<ipython-input-42-08f6854da658>:3: DeprecationWarning: DataFrameGroupBy.apply ope
rated on the grouping columns. This behavior is deprecated, and in a future versi
on of pandas the grouping columns will be excluded from the operation. Either pas
s `include_groups=False` to exclude the groupings or explicitly select the groupi
ng columns after groupby to silence this warning.
  top2_positions = pos_ctc.groupby('company_hash').apply(lambda x: x.sort_values
(by='ctc',ascending=False).head(2)).reset_index(drop=True)
```

In [47]:
```
#  Data preprocessing for Unsupervised Clustering

# Select features for clustering
features = ['company_hash', 'job_position', 'years_of_experience', 'ctc']

# Encoding categorical variables
le_company = LabelEncoder()
le_job = LabelEncoder()

df['company_enc'] = le_company.fit_transform(df['company_hash'])
df['job_enc'] = le_job.fit_transform(df['job_position'])

X = df[['company_enc', 'job_enc', 'years_of_experience', 'ctc']]
X.head(10)
```

Out[47]:

	company_enc	job_enc	years_of_experience	ctc
0	969	455	9	1100000
1	19730	289	7	449999
2	15512	138	10	2000000
3	12108	138	8	700000
4	20226	289	8	1400000
5	35570	289	7	700000
6	10162	289	7	1500000
7	29158	138	6	400000
8	25405	422	5	450000
9	33129	422	6	360000

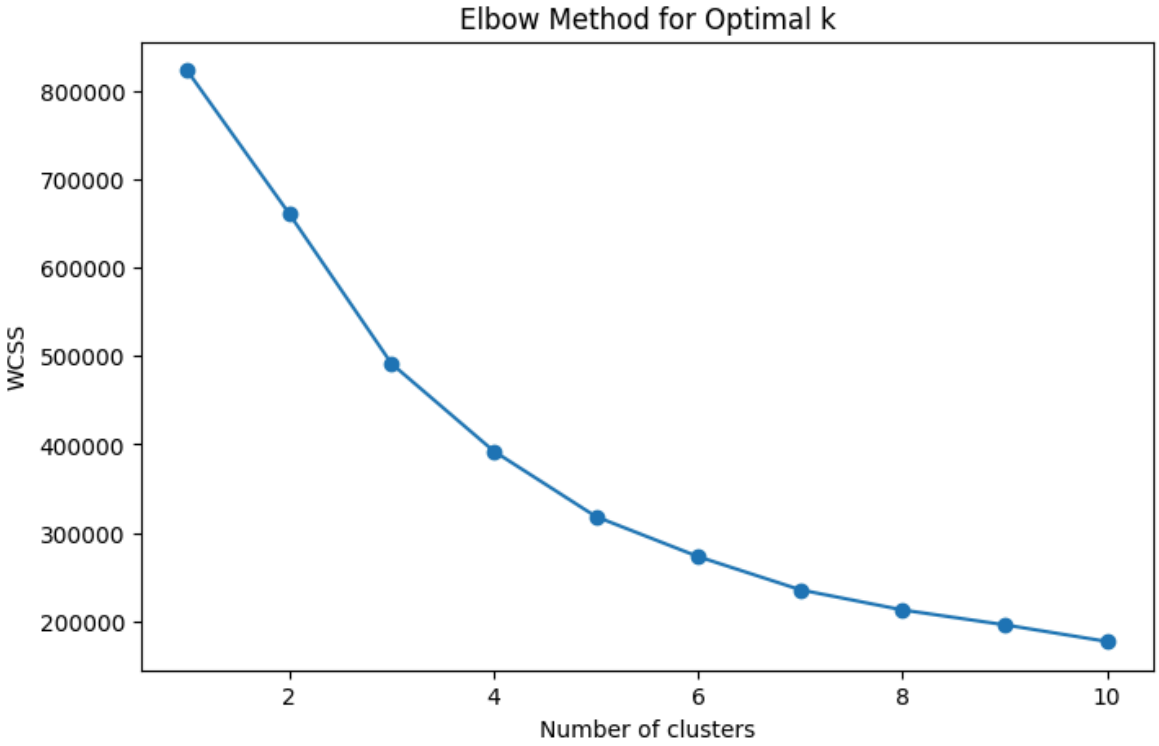
In [48]:
```
# Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [49]:
```
 # Check clustering tendency (Hopkins statistic)


from sklearn.neighbors import NearestNeighbors
import random

def hopkins(X):
    d = X.shape[1]
    n = len(X)  # rows
    m = int(0.1 * n)  # sample size 10%
    nbrs = NearestNeighbors(n_neighbors=1).fit(X)
    rand_X = np.random.uniform(np.min(X, axis=0), np.max(X, axis=0), (m, d))
    ujd = []
    wjd = []
    for j in range(m):
        u_dist, _ = nbrs.kneighbors([rand_X[j]], 2, return_distance=True)
        ujd.append(u_dist[0][1])
        w_dist, _ = nbrs.kneighbors([X[random.randint(0, n-1)]], 2, return_dista
        wjd.append(w_dist[0][1])
    H = sum(ujd) / (sum(ujd) + sum(wjd))
    return H

hopkins_stat = hopkins(X_scaled)
print(f"Hopkins statistic: {hopkins_stat:.3f}")
# If close to 1, data is clusterable
```

```
Hopkins statistic: 0.999
```

In [50]:
```
# Elbow method to find optimal k for KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)
```

```
plt.figure(figsize=(8,5))
plt.plot(range(1, 11), wcss, marker='o')
plt.title('Elbow Method for Optimal k')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

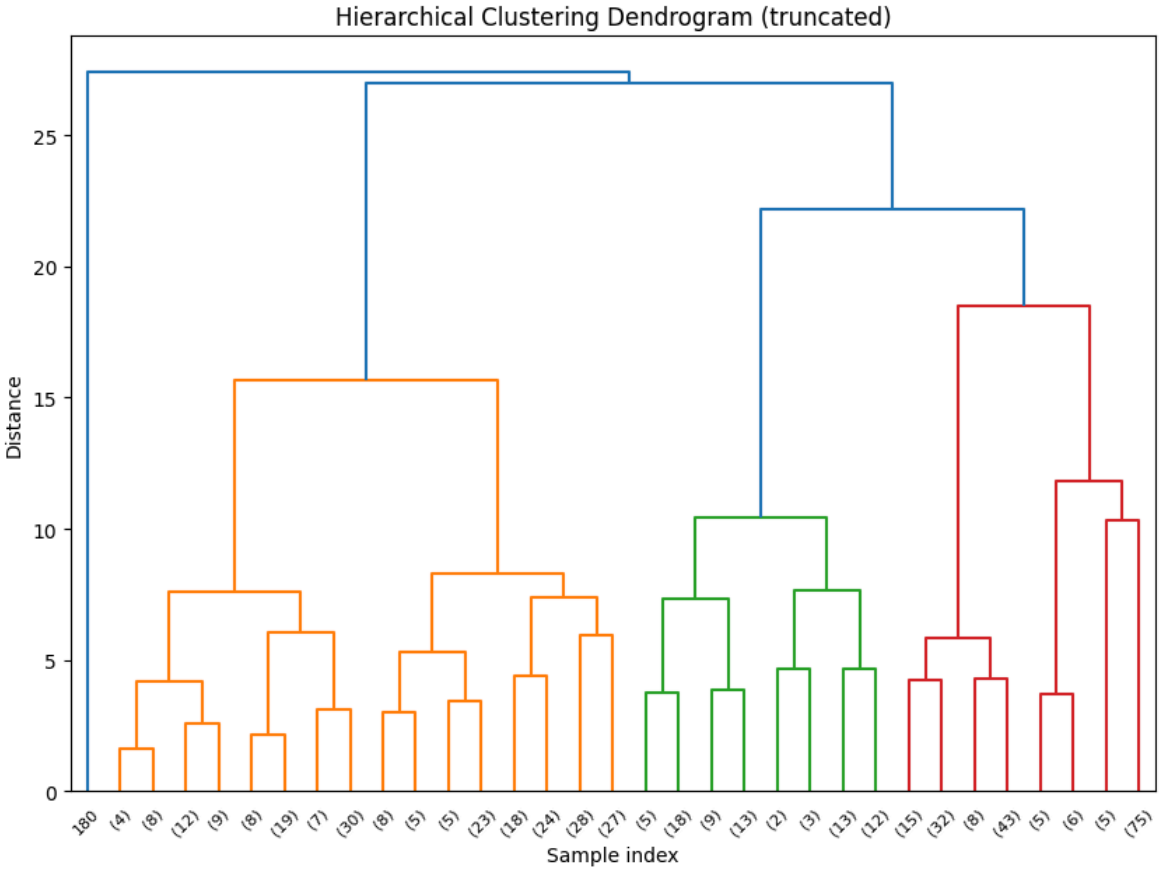


In [52]:
```
# K-means clustering with chosen k (say k=4 based on elbow)
k = 4
kmeans = KMeans(n_clusters=k, random_state=42)
df['KMeans_Cluster'] = kmeans.fit_predict(X_scaled)

# Cluster profile summary
print(df.groupby('KMeans_Cluster')[['ctc', 'years_of_experience']].mean())
```

```
                        ctc  years_of_experience
KMeans_Cluster
0              1.277489e+06             8.574136
1              2.228639e+06            17.439703
2              1.351530e+08             9.548653
3              1.235777e+06             8.511711
```

In [54]:
```
# Hierarchical clustering (on a sample if large)
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster

sample_df = df.sample(n=500, random_state=42)
sample_X = sample_df[['company_enc', 'job_enc', 'years_of_experience', 'ctc']]
sample_X_scaled = scaler.fit_transform(sample_X)

linked = linkage(sample_X_scaled, method='ward')

plt.figure(figsize=(10, 7))
dendrogram(linked, truncate_mode='level', p=5)
plt.title('Hierarchical Clustering Dendrogram (truncated)')
plt.xlabel('Sample index')
plt.ylabel('Distance')
```

```
plt.show()

# Assign cluster labels from hierarchical clustering (e.g., 4 clusters)
sample_df['Hier_Cluster'] = fcluster(linked, 4, criterion='maxclust')

print(sample_df.groupby('Hier_Cluster')[['ctc', 'years_of_experience']].mean())
```



Hier_Cluster	ctc	years_of_experience
1	1.128447e+06	9.004255
2	2.087867e+06	17.746667
3	1.390753e+06	8.455026
4	6.060000e+07	10.000000