Core Integra Talent Acquisition Performance

August 23, 2024

1 Core Integra Talent Acquisition Performance

1.1 Project Description

Core Integra offers the service of candidate selection and screening for clients through their talent acquisition team. In this project, these hiring processes funnels will be analyzed with the goal of optimizing processes such as candidate selection, average time between assignment and joining date and more. This analysis will also provide detailed Key Performance Indicators for each employee part of the Talent Acquisition team at Core Integra, which can then be used in order to track individual performance and assign goals for each employee. Lastly, this project will also explore why and how to implement better data collection processes in order to improve the accuracy of the analysis for the future.

1.1.1 1. Loading & Exploring the Dataset

[3]:
```
import pandas as pd
import numpy

df=pd.read_csv("TCS_MIS.csv")
df.head()
```

[3]:
```
   Sr.NO Assignment Date Client            Position  \
0      1          1/2/24    TCS  TCS - Finance SPOC
1      2          1/2/24    TCS  TCS - Finance SPOC
2      3          1/2/24    TCS  TCS - Finance SPOC
3      4          1/2/24    TCS  TCS - Finance SPOC
4      5          1/2/24    TCS  TCS - Finance SPOC

                  Candidate Name Contact No. EmailID Current Location  \
0                   Shiny Tavero  8879536187     NaN           Kamote
1           Mayuri Dilip Bulunge  9561559680     NaN      Navi Mumbai
2         Anuja Prashant Nakhawa  9082610082     NaN             Uran
3  Pranali Pandharinath Padelkar  9619375762     NaN         Kharghar
4          Rupali Baburao Warang  8847785920     NaN      Navi Mumbai

         State  Qualification  … Profile shared on  Interviwe Date   \
0  Maharashtra           NaN  …             1/2/24          1/3/24
1  Maharashtra           NaN  …             1/2/24             NaN
```

```
2  Maharashtra           NaN  …              1/2/24             NaN
3  Maharashtra           NaN  …              1/2/24             NaN
4  Maharashtra           NaN  …              1/2/24             NaN

  Tel / F2F Interview Date - Feedback date Client Feedback  \
0                                   1/3/24         No Show
1                                      NaN             NaN
2                                      NaN             NaN
3                                      NaN             NaN
4                                      NaN             NaN

  Documentation status Offer Status Joining Status  DOJ Joining Month  \
0                  NaN          NaN             NaN  NaN           NaN
1                  NaN          NaN             NaN  NaN           NaN
2                  NaN          NaN             NaN  NaN           NaN
3                  NaN          NaN             NaN  NaN           NaN
4                  NaN          NaN             NaN  NaN           NaN

          Spoc Name
0  Anisha Bhaskaran
1  Anisha Bhaskaran
2  Anisha Bhaskaran
3  Anisha Bhaskaran
4  Anisha Bhaskaran

[5 rows x 24 columns]
```

[5]:
```python
# Display column names
print("Column names:")
print(df.columns)

# Display data types of each column
print("\nData types:")
print(df.dtypes)
```

```
Column names:
Index(['Sr.NO', 'Assignment Date', 'Client', 'Position', 'Candidate Name',
       'Contact No.', 'EmailID', 'Current Location', 'State ', 'Qualification',
       'HR', 'CI HR Status', 'Additional Remark - Executive',
       'Reference given by', 'Profile shared on ', 'Interviwe Date ',
       'Tel / F2F Interview Date - Feedback date', 'Client Feedback',
       'Documentation status', 'Offer Status', 'Joining Status', 'DOJ',
       'Joining Month', 'Spoc Name'],
      dtype='object')

Data types:
Sr.NO                                    int64
```

```
Assignment Date                                 object
Client                                          object
Position                                        object
Candidate Name                                  object
Contact No.                                     object
EmailID                                         object
Current Location                                object
State                                           object
Qualification                                   object
HR                                              object
CI HR Status                                    object
Additional Remark - Executive                   object
Reference given by                              object
Profile shared on                               object
Interviwe Date                                  object
Tel / F2F Interview Date - Feedback date        object
Client Feedback                                 object
Documentation status                            object
Offer Status                                    object
Joining Status                                  object
DOJ                                             object
Joining Month                                   object
Spoc Name                                       object
dtype: object
```

[7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8346 entries, 0 to 8345
Data columns (total 24 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Sr.NO                                   8346 non-null   int64
 1   Assignment Date                         8346 non-null   object
 2   Client                                  8346 non-null   object
 3   Position                                8346 non-null   object
 4   Candidate Name                          8346 non-null   object
 5   Contact No.                             8346 non-null   object
 6   EmailID                                 5117 non-null   object
 7   Current Location                        8346 non-null   object
 8   State                                   8346 non-null   object
 9   Qualification                           3176 non-null   object
 10  HR                                      8346 non-null   object
 11  CI HR Status                            8346 non-null   object
 12  Additional Remark - Executive           8346 non-null   object
 13  Reference given by                      8223 non-null   object
 14  Profile shared on                       3388 non-null   object
 15  Interviwe Date                          1635 non-null   object
```

```
 16  Tel / F2F Interview Date - Feedback date  1650 non-null   object
 17  Client Feedback                           2595 non-null   object
 18  Documentation status                      348 non-null    object
 19  Offer Status                              315 non-null    object
 20  Joining Status                            299 non-null    object
 21  DOJ                                       214 non-null    object
 22  Joining Month                             171 non-null    object
 23  Spoc Name                                 8346 non-null   object
dtypes: int64(1), object(23)
memory usage: 1.5+ MB
```

1.1.2 2. Data Cleaning

[9]:
```
df = df.drop_duplicates(subset=['Candidate Name'])
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7752 entries, 0 to 8345
Data columns (total 24 columns):
 #   Column                                    Non-Null Count  Dtype
---  ------                                    --------------  -----
 0   Sr.NO                                     7752 non-null   int64
 1   Assignment Date                           7752 non-null   object
 2   Client                                    7752 non-null   object
 3   Position                                  7752 non-null   object
 4   Candidate Name                            7752 non-null   object
 5   Contact No.                               7752 non-null   object
 6   EmailID                                   4817 non-null   object
 7   Current Location                          7752 non-null   object
 8   State                                     7752 non-null   object
 9   Qualification                             2972 non-null   object
 10  HR                                        7752 non-null   object
 11  CI HR Status                              7752 non-null   object
 12  Additional Remark - Executive             7752 non-null   object
 13  Reference given by                        7659 non-null   object
 14  Profile shared on                         3111 non-null   object
 15  Interviwe Date                            1478 non-null   object
 16  Tel / F2F Interview Date - Feedback date  1492 non-null   object
 17  Client Feedback                           2348 non-null   object
 18  Documentation status                      310 non-null    object
 19  Offer Status                              288 non-null    object
 20  Joining Status                            276 non-null    object
 21  DOJ                                       197 non-null    object
 22  Joining Month                             155 non-null    object
 23  Spoc Name                                 7752 non-null   object
dtypes: int64(1), object(23)
memory usage: 1.5+ MB
```

[11]:
```
df['Assignment Date'] = pd.to_datetime(df['Assignment Date'], errors='coerce')
df['DOJ'] = pd.to_datetime(df['DOJ'], errors='coerce')
df['Interviwe Date '] = pd.to_datetime(df['Interviwe Date '], errors='coerce')
df.head()
```

```
/tmp/ipykernel_126/3122570799.py:1: UserWarning: Could not infer format, so each
element will be parsed individually, falling back to `dateutil`. To ensure
parsing is consistent and as-expected, please specify a format.
  df['Assignment Date'] = pd.to_datetime(df['Assignment Date'], errors='coerce')
/tmp/ipykernel_126/3122570799.py:2: UserWarning: Could not infer format, so each
element will be parsed individually, falling back to `dateutil`. To ensure
parsing is consistent and as-expected, please specify a format.
  df['DOJ'] = pd.to_datetime(df['DOJ'], errors='coerce')
/tmp/ipykernel_126/3122570799.py:3: UserWarning: Could not infer format, so each
element will be parsed individually, falling back to `dateutil`. To ensure
parsing is consistent and as-expected, please specify a format.
  df['Interviwe Date '] = pd.to_datetime(df['Interviwe Date '], errors='coerce')
```

[11]:

	Sr.NO	Assignment Date	Client	Position	\
0	1	2024-01-02	TCS	TCS - Finance SPOC	
1	2	2024-01-02	TCS	TCS - Finance SPOC	
2	3	2024-01-02	TCS	TCS - Finance SPOC	
3	4	2024-01-02	TCS	TCS - Finance SPOC	
4	5	2024-01-02	TCS	TCS - Finance SPOC	

	Candidate Name	Contact No.	EmailID	Current Location	\
0	Shiny Tavero	8879536187	NaN	Kamote	
1	Mayuri Dilip Bulunge	9561559680	NaN	Navi Mumbai	
2	Anuja Prashant Nakhawa	9082610082	NaN	Uran	
3	Pranali Pandharinath Padelkar	9619375762	NaN	Kharghar	
4	Rupali Baburao Warang	8847785920	NaN	Navi Mumbai	

	State	Qualification	…	Profile shared on	Interviwe Date	\
0	Maharashtra	NaN	…	1/2/24	2024-01-03	
1	Maharashtra	NaN	…	1/2/24	NaT	
2	Maharashtra	NaN	…	1/2/24	NaT	
3	Maharashtra	NaN	…	1/2/24	NaT	
4	Maharashtra	NaN	…	1/2/24	NaT	

	Tel / F2F Interview Date - Feedback date	Client Feedback	\
0	1/3/24	No Show	
1	NaN	NaN	
2	NaN	NaN	
3	NaN	NaN	
4	NaN	NaN	

	Documentation status	Offer Status	Joining Status	DOJ	Joining Month	\

```
0                  NaN          NaN             NaN NaT           NaN
1                  NaN          NaN             NaN NaT           NaN
2                  NaN          NaN             NaN NaT           NaN
3                  NaN          NaN             NaN NaT           NaN
4                  NaN          NaN             NaN NaT           NaN

          Spoc Name
0  Anisha Bhaskaran
1  Anisha Bhaskaran
2  Anisha Bhaskaran
3  Anisha Bhaskaran
4  Anisha Bhaskaran

[5 rows x 24 columns]
```

1.1.3 3. Creating New Columns

Note: In this next step, we will create dummy variables to convert text from columns into numerical values, in order to help with future analysis. For example, in this next step we will count how many rows in the 'Joining Status' column are set to 'Joined' and convert those entries into binary results (1 if Joined and 0 if not). We will do a similar trick for the other columns to understand how many candidates received offers etc.

[13]:
```
df2=pd.get_dummies(df['Joining Status'])
df3=pd.get_dummies(df['CI HR Status'])
df4=pd.get_dummies(df['Client Feedback'])
df5=pd.get_dummies(df['Documentation status'])
df6=pd.get_dummies(df['Offer Status'])

df['Joined']=df2['Joined'].astype(int)
df['CI HR Interested']=df3['Interested'].astype(int)
df['Client Selected']=df4['Selected'].astype(int)
df['Documentation Done']=df5['Done'].astype(int)
df['Offer Done']=df6['Done '].astype(int)
df['Assignment to Joining Date'] = (df['DOJ'] - df['Assignment Date']).dt.days
df['Assignment to Interview Date'] = (df['Interviwe Date '] - df['Assignment␣
 ↪Date']).dt.days
df['Interview Date to Joining Date'] = (df['DOJ'] - df['Interviwe Date ']).dt.
 ↪days

df.head()
```

[13]:
```
   Sr.NO Assignment Date Client            Position  \
0      1      2024-01-02    TCS  TCS - Finance SPOC
1      2      2024-01-02    TCS  TCS - Finance SPOC
2      3      2024-01-02    TCS  TCS - Finance SPOC
3      4      2024-01-02    TCS  TCS - Finance SPOC
```

```
4      5      2024-01-02    TCS  TCS - Finance SPOC

                 Candidate Name Contact No. EmailID Current Location  \
0                   Shiny Tavero  8879536187     NaN           Kamote
1           Mayuri Dilip Bulunge  9561559680     NaN      Navi Mumbai
2         Anuja Prashant Nakhawa  9082610082     NaN             Uran
3  Pranali Pandharinath Padelkar  9619375762     NaN         Kharghar
4          Rupali Baburao Warang  8847785920     NaN      Navi Mumbai

        State  Qualification  … Joining Month         Spoc Name Joined  \
0  Maharashtra           NaN  …           NaN  Anisha Bhaskaran      0
1  Maharashtra           NaN  …           NaN  Anisha Bhaskaran      0
2  Maharashtra           NaN  …           NaN  Anisha Bhaskaran      0
3  Maharashtra           NaN  …           NaN  Anisha Bhaskaran      0
4  Maharashtra           NaN  …           NaN  Anisha Bhaskaran      0

  CI HR Interested Client Selected Documentation Done Offer Done  \
0                1               0                  0          0
1                0               0                  0          0
2                0               0                  0          0
3                0               0                  0          0
4                0               0                  0          0

  Assignment to Joining Date Assignment to Interview Date  \
0                        NaN                          1.0
1                        NaN                          NaN
2                        NaN                          NaN
3                        NaN                          NaN
4                        NaN                          NaN

  Interview Date to Joining Date
0                            NaN
1                            NaN
2                            NaN
3                            NaN
4                            NaN

[5 rows x 32 columns]
```

1.1.4 4. Improving Data Quality

Note: In this next step, we want to remove the first 6 characters in the 'Position' column, as the information about the client is redundant due to the previous column (Client).

[15]:
```
import pandas as pd

df['Position'] = df['Position'].str[6:]
```

```
df.head()
```

[15]:

	Sr.NO	Assignment Date	Client	Position	Candidate Name	\
0	1	2024-01-02	TCS	Finance SPOC	Shiny Tavero	
1	2	2024-01-02	TCS	Finance SPOC	Mayuri Dilip Bulunge	
2	3	2024-01-02	TCS	Finance SPOC	Anuja Prashant Nakhawa	
3	4	2024-01-02	TCS	Finance SPOC	Pranali Pandharinath Padelkar	
4	5	2024-01-02	TCS	Finance SPOC	Rupali Baburao Warang	

	Contact No.	EmailID	Current Location	State	Qualification	…	\
0	8879536187	NaN	Kamote	Maharashtra	NaN	…	
1	9561559680	NaN	Navi Mumbai	Maharashtra	NaN	…	
2	9082610082	NaN	Uran	Maharashtra	NaN	…	
3	9619375762	NaN	Kharghar	Maharashtra	NaN	…	
4	8847785920	NaN	Navi Mumbai	Maharashtra	NaN	…	

	Joining Month	Spoc Name	Joined CI HR	Interested Client Selected	\
0	NaN	Anisha Bhaskaran	0	1	0
1	NaN	Anisha Bhaskaran	0	0	0
2	NaN	Anisha Bhaskaran	0	0	0
3	NaN	Anisha Bhaskaran	0	0	0
4	NaN	Anisha Bhaskaran	0	0	0

	Documentation Done	Offer Done	Assignment to Joining Date	\
0	0	0	NaN	
1	0	0	NaN	
2	0	0	NaN	
3	0	0	NaN	
4	0	0	NaN	

	Assignment to Interview Date	Interview Date to Joining Date
0	1.0	NaN
1	NaN	NaN
2	NaN	NaN
3	NaN	NaN
4	NaN	NaN

[5 rows x 32 columns]

[196]:
```
df['Qualification'].value_counts().head(40)
```

[196]:

Qualification	
nan	4780
Graduate	458
MBA	264
B.COM	253

```
Bcom                        215
B. Com                      210
BCA                         147
B. A.                       134
B. Sc                       129
BMS                          90
BA                           85
B.TECH                       80
Bsc                          80
B.SC                         71
BBA                          63
B.A                          56
BE                           39
Mcom                         37
Btech                        35
MCA                          32
12th                         26
Under Graduate               22
Msc                          19
M.COM                        18
MA                           17
MSW                          16
B. Tech                      16
CA                           15
M. Com                       15
M.A                          15
BE / B.Tech / BTech          15
M.SC                         13
Army - Graduation            12
Army - Graduate              12
BSC                          11
M. Sc                        10
Diploma                       9
HSC                           8
M. A.                         8
PG                            7
Name: count, dtype: int64
```

Next, we want to clean the Qualification column better so that we can run analysis on how a qualification of a candidates affects their chances of getting hired. We will do this by creating another column - Qualification Category - which will categorize all of the entries in the Qualification column.

[19]:
```
df['Qualification']=df['Qualification'].astype(str)

def classify_qualification(qualification):
```

```
    if pd.isna(qualification) or qualification == '' or str(qualification).
↪strip().lower() == 'nan':
        return 'Blank Entry'

    qualification = qualification.strip().lower()

    if 'b.com' in qualification or 'bcom' in qualification:
        return 'BCom'
    elif 'b.a' in qualification or 'ba' in qualification or 'bfa' in␣
↪qualification or 'baf' in qualification:
        return 'BA'
    elif 'b.sc' in qualification or 'bsc' in qualification:
        return 'BSc'
    elif 'b.tech' in qualification or 'btech' in qualification or 'be' in␣
↪qualification:
        return 'BTech/BE'
    elif 'bba' in qualification or 'bms' in qualification:
        return 'BBA/BMS'
    elif 'mba' in qualification:
        return 'MBA'
    elif 'm.com' in qualification or 'mcom' in qualification:
        return 'MCom'
    elif 'm.sc' in qualification or 'msc' in qualification:
        return 'MSc'
    elif 'm.a' in qualification or 'ma' in qualification:
        return 'MA'
    elif 'diploma' in qualification or 'pg' in qualification or 'post graduate'␣
↪in qualification:
        return 'Diploma/PG'
    elif '12th' in qualification or 'hsc' in qualification or 'high school' in␣
↪qualification:
        return '12th/High School'
    else:
        return 'Other'

df['Qualification Category'] = df['Qualification'].apply(classify_qualification)

df['Qualification Category'].value_counts()
```

[19]:
```
Qualification Category
Blank Entry         4780
Other               1311
BA                   518
BCom                 483
BTech/BE             185
BSc                  185
BBA/BMS               90
```

```
MA                    58
MCom                  58
MSc                   38
12th/High School      36
Diploma/PG            10
Name: count, dtype: int64
```

Therefore, as seen above, the Qualification Categories have been classified better, although there are still 4780 Blank Entries.

Next, let's look at the 'Reference given by' column and try to classify that into Reference Categories.

[21]:
```
df['Reference given by'].value_counts()
```

[21]:
```
Reference given by
Naukri                                4668
NAUKRI                                 824
Job Hai                                456
Skill Connect                          340
Surender Diwedi                        209
Naukri- Mass Email                     135
Candidate Referance                    126
Local Referance                        107
Captain Tobby Joseph                    84
Ex-Servicemen                           71
Saloni                                  58
Dhruthi                                 53
Naukri - Mass Email                     44
Swastik                                 38
Tescom                                  32
NAUKRI JP                               32
Client Referance - Dipti                30
SURENDRA                                30
Client Referance                        23
HR Mecatric                             19
Naukri JP                               18
Skill C2C                               17
Temp Pull Data                          14
Work India                              14
Mukesh                                   9
SALONI                                   9
ex-servicemen                            9
JOB POSTING                              8
TCS                                      8
Client Referance - Aditya                8
Ref Candidates                           8
KIRAN                                    7
Anudip Foundation                        6
```

Job Fair	6
Prem	6
WORKINDIA	5
NAUKRI - MASS EMAIL	5
Job India	5
Old Data	4
DHRUTHI	4
surender Diwedi	4
CAPTAIN TOBBY JOSEPH	4
Ranjit	4
Namrata Vendor - Prem	4
Ipsita Pati - Employee Referance	3
Ex-servicemen	3
DGR	3
Lokesh Gupta	3
CLIENT REFERANCE - DIPTI	3
Apna Job	3
Mohit Kumar	3
Reference - Candidate	3
Client Referance - Anshika	2
Client Referance	2
Prasanth Sam	2
Old Data - Mass Email	2
Naukri - Mass email	2
Shweta Patil	2
CANDIDATE REFERANCE	2
Amra Ram Sai	2
Swati C	2
Indrajeet C	2
Linkedin	2
Linkdin	2
TCS Portal	2
ex-servicemen - Air Force	2
Client Referance - Anvi	2
Candidate Referance	2
JOB HAI	2
Own Referance	2
SURNDRA	2
Sushil Sharma	1
Candidate referance	1
Candidate Reference	1
ex - serviceman - candidate referance	1
Prem	1
WORK INDIA	1
JO	1
NAU	1
SKILL C2C	1

```
JOB                                    1
TESCOM                                 1
EX SERVICEMAN                          1
CA                                     1
Riya Kumari                            1
SURE                                   1
Vinayak                                1
SUDHIR C                               1
Mahesh Sir                             1
Subramanya                             1
Mandar Sarpole                         1
Tushwar Watpade                        1
SURENDER                               1
Candidate referance                    1
Raju                                   1
Anita Vendor                           1
Shubham Patil                          1
Nilesh                                 1
Swapnil Wani                           1
Nitin Katkar                           1
Cadidate Referance                     1
NAVI MUMBAI                            1
Client Referance - Niha                1
Name: count, dtype: int64
```

[137]:
```python
import pandas as pd
import numpy as np

df['Reference given by'] = df['Reference given by'].str.lower()

def classify_reference(ref):
    if pd.isna(ref):  # Check for NaN values
        return 'Other'
    if 'naukri' in ref:
        return 'Naukri'
    elif 'job hai' in ref:
        return 'Job Hai'
    elif 'skill connect' in ref:
        return 'Skill Connect'
    elif 'surender diwedi' in ref or 'surendra' in ref:
        return 'Surender Diwedi'
    elif 'captain toby joseph' in ref or 'captain tobby joseph' in ref:
        return 'Captain Tobby Joseph'
    elif 'ex-servicemen' in ref or 'ex serviceman' in ref:
        return 'Ex-Servicemen'
    elif 'saloni' in ref:
        return 'Saloni'
```

```
    elif 'dhruthi' in ref or 'druthi' in ref:
        return 'Dhruthi'
    elif 'swastik' in ref:
        return 'Swastik'
    elif 'local referance' in ref or 'local reference' in ref:
        return 'Local Reference'
    elif 'candidate referance' in ref or 'candidate reference' in ref:
        return 'Candidate Reference'
    elif 'client referance' in ref or 'client reference' in ref:
        return 'Client Referance'
    else:
        return 'Other'

df['Reference Category'] = df['Reference given by'].apply(classify_reference)

df['Reference Category'].value_counts()
```

[137]:
```
Reference Category
Naukri                  5728
Job Hai                  458
Skill Connect            340
Other                    335
Surender Diwedi          243
Candidate Reference      134
Local Reference          107
Captain Tobby Joseph      88
Ex-Servicemen             86
Client Referance          71
Saloni                    67
Dhruthi                   57
Swastik                   38
Name: count, dtype: int64
```

As you can see, the data is a lot cleaner now and better classified.

1.1.5 5. Calculating Key Performance Indicators

[23]:
```
total_candidates = len(df)
candidates_shortlisted = df['CI HR Interested'].sum()
candidates_selected_by_TCS = df['Client Selected'].sum()
candidates_joined = df['Joined'].sum()
average_time_to_join = df['Assignment to Joining Date'].mean()

print(f"Total Candidates: {total_candidates}")
print(f"Candidates Shortlisted: {candidates_shortlisted}")
print(f"Candidates Selected by TCS: {candidates_selected_by_TCS}")
print(f"Candidates Joined: {candidates_joined}")
```

```
print(f"Average Time to Join: {average_time_to_join:.2f} days")
```

```
Total Candidates: 7752
Candidates Shortlisted: 2338
Candidates Selected by TCS: 293
Candidates Joined: 153
Average Time to Join: 34.31 days
```

1.1.6 6. General Analytics

Qualification Analysis of those who joined

[148]:
```
filtered_df = df[df['Client Selected'] == 1]

qualification_counts = filtered_df['Qualification Category'].value_counts()

qualification_counts
```

[148]:
```
Qualification Category
Blank Entry          162
Other                 38
BCom                  29
BA                    26
BSc                   21
BTech/BE               5
MCom                   4
BBA/BMS                3
MSc                    2
MA                     2
12th/High School       1
Name: count, dtype: int64
```

[156]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
qualification_counts.plot(kind='bar', color='skyblue')

plt.title('Qualification Category Distribution for Selected Candidates')
plt.xlabel('Qualification Category')
plt.ylabel('Candidate Count')
plt.xticks(rotation=45, ha='right')

plt.tight_layout()
plt.show()
```

Demographic Analysis

[200]:
```
filtered_df = df[df['Client Selected'] == 1]
filtered_df['State '].value_counts()
```

[200]:
```
State
Maharashtra          127
Uttar Pradesh         47
Bihar                 26
Madhya Pradesh        25
Rajasthan             12
Delhi                 10
Telangana              8
West Bengal            7
Jharkhand              6
Jammu & Kashmir        5
Uttarakhand            5
Andhra Pradesh         3
Gujarat                3
Odisha                 2
Punjab                 2
Karnataka              1
Himachal Pradesh       1
Tamil Nadu             1
Haryana                1
Assam                  1
```

```
Name: count, dtype: int64
```

Positions that have Joined TCS

[163]:
```
filtered_df = df[df['Joined'] == 1]
filtered_df['Position'].value_counts()
```

[163]:
```
Position
OE                                  45
VADM                                43
DSE                                 12
SCE Thane                            8
ASS/ATS                              7
Regional HR                          6
Associate                            4
Transformation Manager               4
Support Operations Executive         3
Platform Tester                      3
Admin Executive                      3
SO                                   2
Lead EF Resource Management          2
CCTV Executive                       2
SME                                  1
Data Analyst                         1
Telesales Executive                  1
MIS Executive                        1
Technical Support                    1
Developer                            1
Finance SPOC                         1
ASM                                  1
Translation Project coordinator      1
Name: count, dtype: int64
```

Reference Category Analysis

[173]:
```
filtered_df = df[df['Client Selected'] == 1]
reference_counts = filtered_df['Reference Category'].value_counts()
reference_counts
```

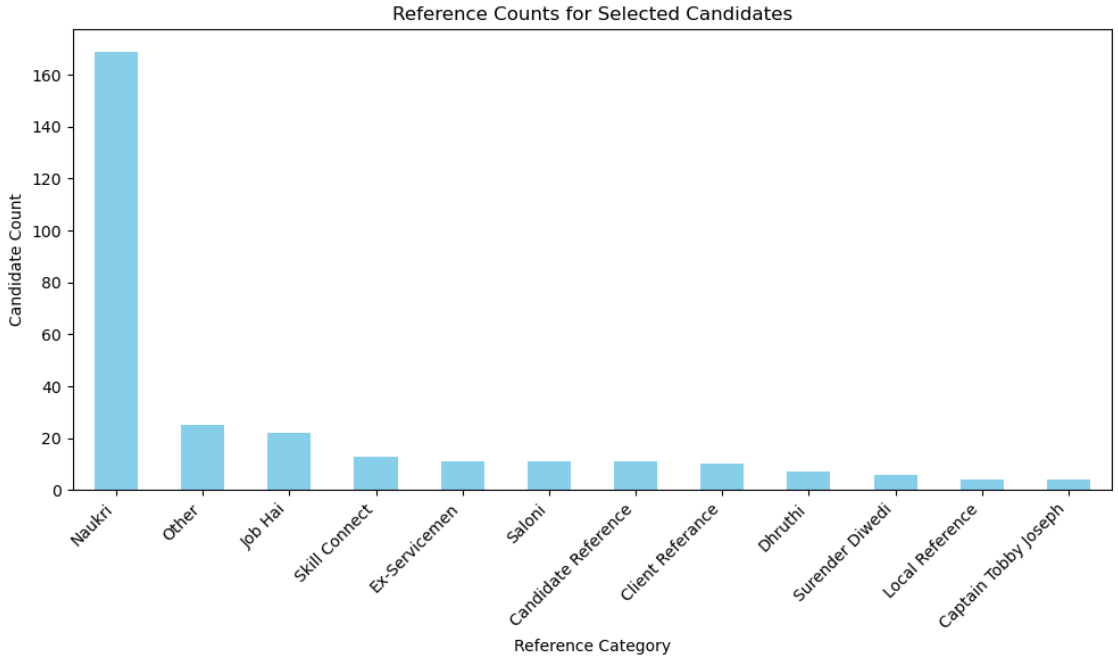
[173]:
```
Reference Category
Naukri                  169
Other                    25
Job Hai                  22
Skill Connect            13
Ex-Servicemen            11
Saloni                   11
Candidate Reference      11
Client Referance         10
```

```
Dhruthi                   7
Surender Diwedi           6
Local Reference           4
Captain Tobby Joseph      4
Name: count, dtype: int64
```

[175]:
```
plt.figure(figsize=(10, 6))
reference_counts.plot(kind='bar', color='skyblue')

plt.title('Reference Counts for Selected Candidates')
plt.xlabel('Reference Category')
plt.ylabel('Candidate Count')
plt.xticks(rotation=45, ha='right')

plt.tight_layout()
plt.show()
```



1.1.7 7. Core Integra Talent Acquisition Team Performance

In this section, I will be exploring different methods and techniques to analyze the data, such as data visualization tools, regression analysis, and more, to extract key insights.

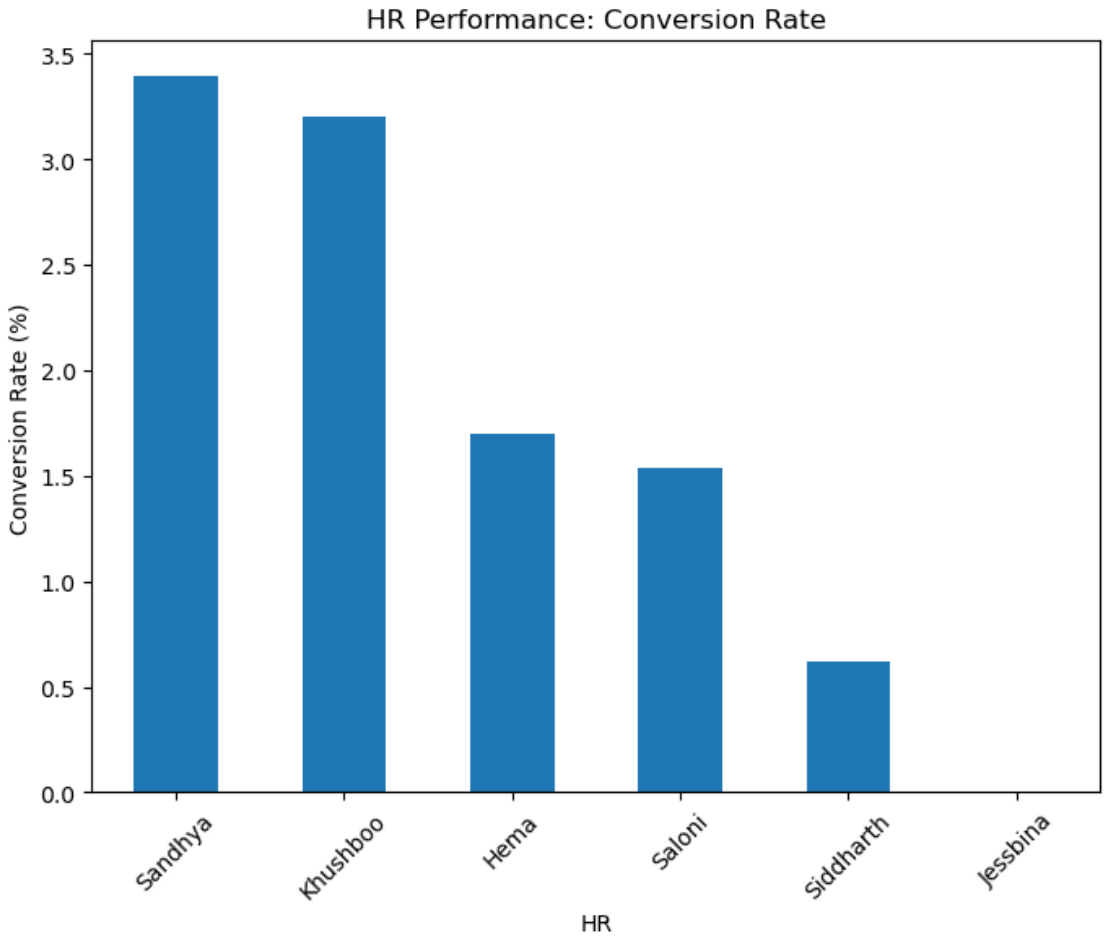
[25]:
```
import matplotlib.pyplot as plt
import seaborn as sns

total_candidates = df.groupby('HR')['Candidate Name'].count()
```

```
joined_candidates = df.groupby('HR')['Joined'].sum()

conversion_rate = (joined_candidates / total_candidates) * 100

plt.figure(figsize=(8, 6))
conversion_rate.sort_values(ascending=False).plot(kind='bar')
plt.title('HR Performance: Conversion Rate')
plt.xlabel('HR')
plt.ylabel('Conversion Rate (%)')
plt.xticks(rotation=45)
plt.show()
```



[27]:
```
hr_grouped = df.groupby('HR').agg(
    total_candidates=('HR', 'count'),
    total_joined=('Joined', 'sum'),
)
```

```
hr_grouped['conversion_rate'] = (hr_grouped['total_joined'] /␣
 ↪hr_grouped['total_candidates']) * 100

hr_grouped = hr_grouped.sort_values(by='conversion_rate', ascending=False)

hr_grouped
```

[27]:

HR	total_candidates	total_joined	conversion_rate
Sandhya	1386	47	3.391053
Khushboo	1250	40	3.200000
Hema	1649	28	1.697999
Saloni	1825	28	1.534247
Siddharth	1619	10	0.617665
Jessbina	23	0	0.000000

Insight: As seen in the above table and graph, it is clear that Sandhya has had the most number of candidates that have joined the client, and the highest conversion rate in doing so.

However, we could analyze this a bit further, and try to see why these employees have such low conversion rates by exploring the entire hiring process funnel.

[29]:
```
hr_grouped_detailed = df.groupby('HR').agg(
    total_candidates=('HR', 'count'),
    total_screened=('CI HR Interested', 'sum'),
    total_selected_by_TCS=('Client Selected', 'sum'),
    total_joined=('Joined', 'sum'),
)

hr_grouped_detailed['screened %'] = (hr_grouped_detailed['total_screened'] /␣
 ↪hr_grouped_detailed['total_candidates']) * 100
hr_grouped_detailed['selected %'] =␣
 ↪(hr_grouped_detailed['total_selected_by_TCS'] /␣
 ↪hr_grouped_detailed['total_screened']) * 100
hr_grouped_detailed['joined %'] = (hr_grouped_detailed['total_joined'] /␣
 ↪hr_grouped_detailed['total_selected_by_TCS']) * 100

# Round specific columns to two decimal places
hr_grouped_detailed['screened %'] = hr_grouped_detailed['screened %'].round(2)
hr_grouped_detailed['selected %'] = hr_grouped_detailed['selected %'].round(2)
hr_grouped_detailed['joined %'] = hr_grouped_detailed['joined %'].round(2)

# Reorder columns
hr_grouped_detailed = hr_grouped_detailed[
    ['total_candidates', 'screened %', 'total_screened', 'selected %',␣
 ↪'total_selected_by_TCS', 'joined %', 'total_joined']
```

```
]

hr_grouped_detailed.sort_values(by='total_joined', ascending=False)
```

[29]:

HR	total_candidates	screened %	total_screened	selected %	\
Sandhya	1386	33.33	462	19.26	
Khushboo	1250	50.64	633	14.38	
Saloni	1825	22.14	404	13.61	
Hema	1649	21.10	348	10.34	
Siddharth	1619	30.14	488	4.51	
Jessbina	23	13.04	3	0.00	

HR	total_selected_by_TCS	joined %	total_joined
Sandhya	89	52.81	47
Khushboo	91	43.96	40
Saloni	55	50.91	28
Hema	36	77.78	28
Siddharth	22	45.45	10
Jessbina	0	NaN	0

In the table above, we have calculated the numbers for the entire hiring funnel, along with their conversion rates. Screened % refers to the percentage of applicants chosen by Core Integra employees. Selected % refers to the percentage of applicants that were selected by the client (TCS) from the ones who were chosen by Core Integra employees. Joined % refers to the percentage of applicants that joined the client after being selected by the client.

Insight: Here, there are some interesting points to note. Khushboo has a very high screened %, which suggests that her selection criteria for candidates might be a bit loose and could be stricter, as from the 633 candidates that Khushboo shortlists, the client has only selected 91, out of which only 40 have joined. Meanwhile, Saloni and Hema have a low screened % which suggests that they need to find better candidates, or loosen their candidate requirements. Despite having a low screened %, Saloni and Hema have a low selected % as well, which suggests that TCS is using a different selection criteria compared to Saloni and Hema.

Next, I want to analyze the reason why Core Integra HR Employees rejected a candidate, in order to better understand their screening processes. Let us first look at the overall reasons for not choosing the 'Interested' option in CI HR Status column i.e. why the Core Integra HR Employee didn't shortlist the candidate. First, we will look at the 'CI HR Status' column, and then the 'Additional Remark - Executive' column in order to get a full understanding of why a client is being rejected. Then, we can look at these entries employee-wise, to understand why each employee is rejecting a candidate.

[31]:
```
df['CI HR Status'].value_counts()
```

[31]:
```
CI HR Status
Interested                              2338
```

```
Not Interested                          1937
Not Connected                           1578
Rejected                                 756
Call Back                                377
NA Undergraduate                         239
Reconnected                              219
Salary Issue                             140
Location Issue                           111
Interested But Vehicle Not Available      19
Not interested                            17
Call back                                 14
Position on Hold                           2
NA B.E. Candidate                          2
Location issue                             2
rejected                                   1
Name: count, dtype: int64
```

[33]:
```
exclude_statuses = ['Interested', 'Not Interested', 'Rejected', 'interested']
df_filtered = df[~df['CI HR Status'].isin(exclude_statuses)]

grouped = df_filtered.groupby(['HR', 'CI HR Status']).size().
 ↪reset_index(name='count')

top5_per_hr = grouped.groupby('HR').apply(lambda x: x.nlargest(5, 'count')).
 ↪reset_index(drop=True)

pivot_table = top5_per_hr.pivot(index='HR', columns='CI HR Status',␣
 ↪values='count').fillna(0)

pivot_table
```

[33]:
```
CI HR Status  Call Back  Call back  Location Issue  NA Undergraduate  \
HR
Hema               93.0        0.0            46.0              16.0
Jessbina            9.0        0.0             1.0               0.0
Khushboo           21.0        0.0            12.0              96.0
Saloni             72.0        0.0             0.0              69.0
Sandhya           182.0       14.0             0.0               0.0
Siddharth           0.0        0.0            13.0              55.0

CI HR Status  Not Connected  Not interested  Position on Hold  Reconnected  \
HR
Hema                  491.0             0.0               2.0          0.0
Jessbina                0.0             0.0               0.0          0.0
Khushboo              216.0             0.0               0.0          0.0
Saloni                223.0             0.0               0.0        196.0
Sandhya               401.0            17.0               0.0          0.0
```

```
Siddharth             247.0             0.0               0.0          8.0
```

```
CI HR Status  Salary Issue
HR
Hema                   0.0
Jessbina               4.0
Khushboo              36.0
Saloni                62.0
Sandhya               26.0
Siddharth             10.0
```

[35]:
```
df['Additional Remark - Executive'].value_counts()
```

[35]:
```
Additional Remark - Executive
Interested                                2338
Not Responding                            1789
Not looking for Job                        940
No Required Skills                         678
Currently Working                          408
Call Back                                  391
Reason Not Shared                          234
Undergraduate                              229
Not looking for job                        227
Location Issue                             113
No Communication Skills                     76
Exp. above 40K                              64
Looking for Profile other than OE           58
Not Looking for Job                         48
Exp.31K - 40K                               34
Not Interested for Said Role                26
Interested But Vehicle Not Available        19
Exp.26K - 30K                               18
Studying                                    10
Wrong No                                     8
Exp.14K - 20K                                7
Age Above 45                                 7
Looking for Desk Job                         5
Exp.21k to 30k                               5
Exp.21K - 25K                                5
Age Above 50                                 4
Exp.14k - 20K                                4
Exp. above 40k                               3
Position on Hold                             2
B.E. Candidate                               2
Name: count, dtype: int64
```

[37]:
```
exclude_statuses = ['Interested', 'Rejected']
df_filtered = df[~df['Additional Remark - Executive'].isin(exclude_statuses)]

grouped = df_filtered.groupby(['HR', 'Additional Remark - Executive']).size().
 ↪reset_index(name='count')

top5_per_hr = grouped.groupby('HR').apply(lambda x: x.nlargest(5, 'count')).
 ↪reset_index(drop=True)

pivot_table = top5_per_hr.pivot(index='HR', columns='Additional Remark -␣
 ↪Executive', values='count').fillna(0)

pivot_table
```

[37]:

HR	Additional Remark - Executive	Call Back	Currently Working	Exp. above 40K	\
Hema		93.0	0.0	0.0	
Jessbina		9.0	0.0	4.0	
Khushboo		0.0	0.0	0.0	
Saloni		72.0	314.0	0.0	
Sandhya		196.0	0.0	0.0	
Siddharth		0.0	0.0	0.0	

HR	Additional Remark - Executive	Location Issue	\
Hema		0.0	
Jessbina		1.0	
Khushboo		0.0	
Saloni		0.0	
Sandhya		0.0	
Siddharth		13.0	

HR	Additional Remark - Executive	Looking for Profile other than OE	\
Hema		0.0	
Jessbina		0.0	
Khushboo		56.0	
Saloni		0.0	
Sandhya		0.0	
Siddharth		0.0	

HR	Additional Remark - Executive	No Communication Skills	No Required Skills	\
Hema		0.0	206.0	
Jessbina		0.0	1.0	
Khushboo		0.0	0.0	
Saloni		0.0	0.0	

```
Sandhya                                          54.0                60.0
Siddharth                                         0.0               362.0

Additional Remark - Executive  Not Looking for Job  Not Responding  \
HR
Hema                                             0.0           491.0
Jessbina                                         0.0             0.0
Khushboo                                        46.0           220.0
Saloni                                           0.0           419.0
Sandhya                                          0.0           404.0
Siddharth                                        0.0           255.0

Additional Remark - Executive  Not looking for Job  Not looking for job  \
HR
Hema                                            247.0                98.0
Jessbina                                          5.0                 0.0
Khushboo                                         76.0                 0.0
Saloni                                           91.0                 0.0
Sandhya                                         108.0                 0.0
Siddharth                                       413.0                 0.0

Additional Remark - Executive  Reason Not Shared  Undergraduate
HR
Hema                                             0.0           0.0
Jessbina                                         0.0           0.0
Khushboo                                         0.0          96.0
Saloni                                         221.0           0.0
Sandhya                                          0.0           0.0
Siddharth                                        0.0          55.0
```

Next, let us look at the Client Feedback column, with relation to the HR employee who had selected the candidate.

[63]:
```
df['Client Feedback'].value_counts()
```

[63]:
```
Client Feedback
No Turn up            654
Rejected by Client    619
No Show               483
Selected              293
Feedback Awaited      195
No Requirement         65
Closed Internally      18
Position Closed        11
Position on Hold        8
position Closed         2
Name: count, dtype: int64
```

I would like to understand which HR employees are responsible for the high amount of candidates who have not shown up to the interview.

[95]:
```
filtered_df = df[df['Client Feedback'].isin(['No Turn up', 'No Show'])]

hr_responsible = filtered_df.groupby(['HR', 'Client Feedback']).size().
 ↪unstack(fill_value=0)

hr_responsible['Total']=hr_responsible['No Show']+hr_responsible['No Turn up']

hr_responsible
```

[95]:

Client Feedback	No Show	No Turn up	Total
HR			
Hema	64	147	211
Jessbina	0	3	3
Khushboo	168	111	279
Saloni	127	113	240
Sandhya	98	108	206
Siddharth	26	172	198

It is clear to see that a lot of candidates, across all HR employees, are not showing up for their interviews. This could potentially harm Core Integra's standing as they are responsible for the clients, so these numbers would have to be reduced by better communication channels between the candidate, Core Integra, and TCS.

Next, I want to look at the relationship between the Reference Category and No-Shows

[182]:
```
filtered_df = df[df['Client Feedback'].isin(['No Turn up', 'No Show'])]

reference_responsible = filtered_df.groupby(['Reference Category', 'Client␣
 ↪Feedback']).size().unstack(fill_value=0)

reference_responsible['Total']=reference_responsible['No␣
 ↪Show']+reference_responsible['No Turn up']

reference_responsible['Total Candidates'] = df.groupby('Reference␣
 ↪Category')['Candidate Name'].count()
reference_responsible['Total Joined'] = df.groupby('Reference␣
 ↪Category')['Joined'].sum()

reference_responsible.sort_values(by="Total", ascending = False)
```

[182]:

Client Feedback	No Show	No Turn up	Total	Total Candidates	\
Reference Category					
Naukri	327	450	777	5728	
Job Hai	47	48	95	458	
Other	25	35	60	335	

Skill Connect	22	19	41	340
Candidate Reference	21	12	33	134
Captain Tobby Joseph	0	33	33	88
Ex-Servicemen	2	19	21	86
Swastik	9	8	17	38
Saloni	5	12	17	67
Surender Diwedi	5	10	15	243
Local Reference	6	7	13	107
Client Referance	8	1	9	71
Dhruthi	6	0	6	57

Client Feedback	Total Joined
Reference Category	
Naukri	90
Job Hai	9
Other	11
Skill Connect	3
Candidate Reference	8
Captain Tobby Joseph	3
Ex-Servicemen	8
Swastik	0
Saloni	7
Surender Diwedi	3
Local Reference	1
Client Referance	6
Dhruthi	4

Next, we will investigate the total amount of days it takes for each candidate to join TCS, after being assigned to the respective HR representative.

[39]:
```
df_selected = df[['HR', 'Assignment to Interview Date', 'Interview Date to␣
↪Joining Date', 'Assignment to Joining Date']].copy()

averages = df_selected.groupby('HR').mean()

averages = averages.applymap(lambda x: f"{x:.1f}" if isinstance(x, float) else␣
↪x)

print(f"Average Time to Join: {average_time_to_join:.2f} days")

averages.sort_values(by='Assignment to Joining Date')
```

```
Average Time to Join: 34.31 days
```

```
/tmp/ipykernel_126/877351825.py:5: FutureWarning: DataFrame.applymap has been
deprecated. Use DataFrame.map instead.
  averages = averages.applymap(lambda x: f"{x:.1f}" if isinstance(x, float) else
x)
```

[39]:

HR	Assignment to Interview Date	Interview Date to Joining Date	Assignment to Joining Date
Hema	8.4	22.0	20.0
Saloni	15.2	31.0	22.7
Khushboo	11.2	28.2	35.1
Siddharth	4.3	31.4	38.6
Sandhya	12.9	39.3	49.3
Jessbina	nan	nan	nan

Insight: It is clear to see that Hema has the fastest Assignment to Joining time period. However, Hema does not have many candidates that have joined TCS eventually, whereas Sandhya has the most candidates that have joined. Therefore, if Sandhya could bring her average time between assignment and joining date down, then she can become more efficient and improve her conversion rate even further.

Last, we can analyze how each employee is doing position-wise. Are some employees placing candidates in certain positions, while not able to place candidates in other positions? Each position has a different selection criteria, so this would be extremely important to investigate in order to improve the selection criteria.

[45]:
```
df['Position'].value_counts()
```

[45]:

Position	
OE	1727
VADM	1236
Associate	519
Support Operations Executive	498
DSE	490
Transformation Manager	341
Regional HR	323
Business Development Executive / Manager	260
CCTV Executive	232
Finance SPOC	225
SCE Thane	195
Sales Manager	127
ASS/ATS	125
SME	122
ASM	99
GSP Executive	97

Desktop Support Engineer	81
AP Executive	80
MIS Executive	77
ESS Executive	60
SO	60
3D Animator	51
Power Auditor	48
Admin Executive	47
DA Team Member	44
Platform Tester	35
Content Management Executive	34
Java Developer	34
Telesales Executive	31
Support Executive	30
Translation Project coordinator	27
Content Editor	26
Developer	23
Project Manager	22
VBA Developer	20
Network Admin	19
SME Governance	18
Lead Generation Executive	18
SCE	18
Hub Operation Manger	17
Functional Consultant	17
Lead EF Resource Management	16
Jr Legal Counsel	15
Associate SIRT	15
Regional Trainer	14
Support Executive	14
Customer Support	13
Strategic Account	12
Tech Led-Network	12
Support Executive SMTE/Non STEM/Non Domin	11
Python Developer	9
Team Sales Manager	9
Data Analyst	8
Analyst LP Research	7
Operation Executive - 2	6
JAVA Developer	6
Mobile Application Desiganer	4
3D Modeler	4
UI Developer	4
Corporate Manage Service	4
Team Leader	3
Technical Support	3
Pre-Sales Solution Consultant	2

```
Territory Manager                                  2
GSP Executive                                      2
Support Executive SMTE                             2
O E                                                1
Business Analyst VE Sales                          1
Name: count, dtype: int64
```

[47]:
```
grouped = df.groupby(['HR', 'Position']).agg(
    count=('Position', 'size'),
    client_selected=('Client Selected', 'sum'),
    joined=('Joined', 'sum')
).reset_index()

# Step 2: Sort each group by count and get the top 5 positions for each HR
grouped = grouped.sort_values(['HR', 'count'], ascending=[True, False])
top_5_positions = grouped.groupby('HR').head(5)

# Step 3: Restructure the DataFrame
final_df = top_5_positions[['HR', 'Position', 'count', 'client_selected',␣
 ↪'joined']]

# Display the final DataFrame
final_df
```

[47]:

	HR	Position	count	\
20	Hema	Support Operations Executive	463	
24	Hema	Transformation Manager	318	
6	Hema	DSE	169	
25	Hema	VADM	167	
17	Hema	SCE Thane	144	
26	Jessbina	Transformation Manager	23	
35	Khushboo	OE	657	
43	Khushboo	VADM	309	
31	Khushboo	DSE	187	
32	Khushboo	Lead EF Resource Management	16	
36	Khushboo	Platform Tester	16	
54	Saloni	OE	554	
62	Saloni	VADM	296	
46	Saloni	Associate	285	
47	Saloni	CCTV Executive	122	
45	Saloni	ASS/ATS	100	
71	Sandhya	OE	456	
77	Sandhya	VADM	392	
64	Sandhya	Associate	139	
73	Sandhya	Regional HR	124	
65	Sandhya	CCTV Executive	87	
86	Siddharth	Business Development Executive / Manager	254	

```
112  Siddharth                          Sales Manager    127
110  Siddharth                                    SME    122
95   Siddharth                           Finance SPOC    114
109  Siddharth                            Regional HR    100
```

	client_selected	joined
20	4	3
24	8	4
6	7	7
25	2	2
17	8	8
26	0	0
35	45	18
43	26	13
31	12	4
32	2	2
36	5	2
54	6	3
62	21	11
46	9	2
47	1	0
45	7	6
71	43	24
77	28	17
64	3	1
73	3	3
65	10	1
86	1	0
112	4	0
110	1	1
95	2	1
109	2	2

Insight: There is some interesting analysis to be made here, because

1.1.8 8. Client Fulfillment Analysis

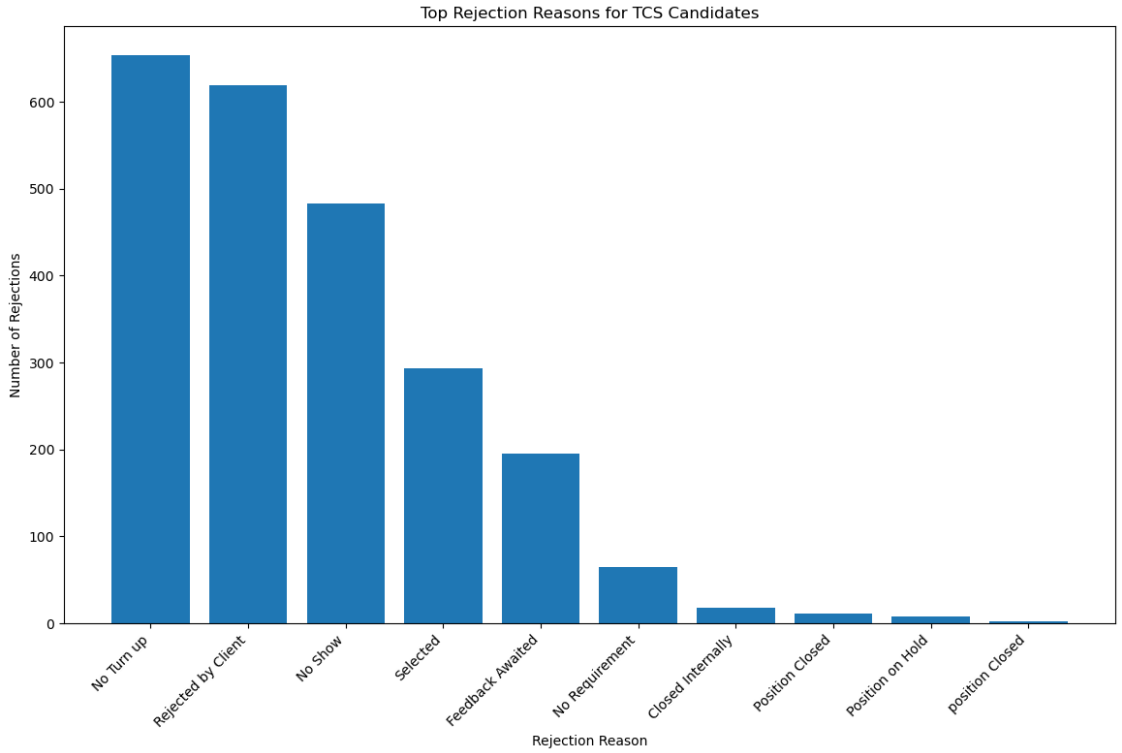
In this section, I will look at some basic analysis, including analyzing the relationship between qualification and client selection, and analyze similar KPIs as written in the above section, but from the client's perspective.

First, we will look at the main reasons for rejection of candidates from the SPOC at TCS.

[51]:
```
tcs_rejection_reasons = df.groupby('Client Feedback').size().
↪reset_index(name='count')

top_tcs_rejection_reasons = tcs_rejection_reasons.sort_values(by='count',␣
↪ascending=False).head(15)
```

```
plt.figure(figsize=(14, 8))
plt.bar(top_tcs_rejection_reasons['Client Feedback'],␣
 ↪top_tcs_rejection_reasons['count'])
plt.title('Top Rejection Reasons for TCS Candidates')
plt.xlabel('Rejection Reason')
plt.ylabel('Number of Rejections')
plt.xticks(rotation=45, ha='right')
plt.show()
```



[53]:
```
top_tcs_rejection_reasons
```

[53]:

	Client Feedback	count
4	No Turn up	654
7	Rejected by Client	619
3	No Show	483
8	Selected	293
1	Feedback Awaited	195
2	No Requirement	65
0	Closed Internally	18
5	Position Closed	11
6	Position on Hold	8
9	position Closed	2

It is clear to see that the main reason candidates are being rejected is due to no shows. There is not enough information to deduce whether this means no shows for an interview, or whether they did not turn up for something else. The stage at which they stop showing up is important to understand.

Next, we can understand further if there is any particular SPOC who has a high number of No Shows. This could be improved by better communication channels between the candidate, the SPOC, and the Core Integra representative.

[125]:
```
filtered_df = df[df['Client Feedback'].isin(['No Turn up', 'No Show'])]

# Group by 'Spoc Name' and count 'No Show' and 'No Turn up'
spoc_responsible = filtered_df.groupby(['Spoc Name', 'Client Feedback']).size().
 ↪unstack(fill_value=0)

# Add a 'Total' column that sums the 'No Show' and 'No Turn up' counts
spoc_responsible['Total'] = spoc_responsible['No Show'] + spoc_responsible['No␣
 ↪Turn up']

# Calculate 'Total Candidates' and 'Total Joined' based on the 'Spoc Name'
spoc_responsible['Total Candidates'] = df.groupby('Spoc Name')['Candidate␣
 ↪Name'].count()
spoc_responsible['Total Joined'] = df.groupby('Spoc Name')['Joined'].sum()

spoc_responsible_filtered = spoc_responsible[spoc_responsible['Total Joined'] >␣
 ↪1]

spoc_responsible_filtered.sort_values(by="Total", ascending=False)
```

[125]:

Client Feedback	No Show	No Turn up	Total	Total Candidates	Total Joined
Spoc Name					
Laxmi Sinha	116	89	205	917	23
Renuka Vadhyar	82	43	125	794	13
Himanshu Dighe	34	33	67	519	4
Ashok Chauhan	14	45	59	250	10
Ravi Kumar Pal	16	37	53	273	12
Giri K	11	27	38	183	2
Ankita Singh	26	12	38	296	13
Vungarala Srikant	15	23	38	141	6
Riya	31	5	36	346	9
Suraj Kumar	15	16	31	186	5
Nishant Bharadwaj	9	16	25	161	9
Punit Kumar	5	12	17	102	3
Ritu Nanda	10	6	16	521	5
Sonali Kshirsagar	8	6	14	323	6
Jaswant Uppal	7	6	13	102	2
Nitesh Sharma	4	9	13	29	5
Mandar Kapse	11	2	13	133	3

Aditya Narse	5	2	7	93	2
Anvi Gondhali	3	3	6	61	3
Shyamraj Syamalan	0	6	6	46	2
Kiran Kumar	4	2	6	75	3
Jitesh Pise	3	0	3	16	2
Swapnil Wani	0	1	1	27	3

We can see that there are some SPOCs with a high number of total candidates, while maintaining a low 'No Show' rate such as Riya or Ritu Nanda. However, it is important to note that the names with the highest no-shows are also the names with the most candidates and the most candidates joined.

Next, we can see the conversion rate of each SPOC to see who has the best conversion rates for joining candidates.

[89]:
```
spoc_grouped = df.groupby('Spoc Name').agg(
    total_candidates=('HR', 'count'),
    total_joined=('Joined', 'sum'),
)

spoc_grouped['conversion_rate'] = (spoc_grouped['total_joined'] /␣
 ↪spoc_grouped['total_candidates']) * 100

spoc_grouped = spoc_grouped.sort_values(by='conversion_rate', ascending=False)

spoc_filtered = spoc_grouped[spoc_grouped['total_joined']>1]

spoc_filtered.sort_values(by="total_joined", ascending=False)
```

[89]:

Spoc Name	total_candidates	total_joined	conversion_rate
Laxmi Sinha	917	23	2.508179
Renuka Vadhyar	794	13	1.637280
Ankita Singh	296	13	4.391892
Ravi Kumar Pal	273	12	4.395604
Ashok Chauhan	250	10	4.000000
Riya	346	9	2.601156
Nishant Bharadwaj	161	9	5.590062
Sonali Kshirsagar	323	6	1.857585
Vungarala Srikant	141	6	4.255319
Suraj Kumar	186	5	2.688172
Nitesh Sharma	29	5	17.241379
Ritu Nanda	521	5	0.959693
Himanshu Dighe	519	4	0.770713
Swapnil Wani	27	3	11.111111
Anvi Gondhali	61	3	4.918033
Kiran Kumar	75	3	4.000000
Mandar Kapse	133	3	2.255639

Punit Kumar	102	3	2.941176
Jitesh Pise	16	2	12.500000
Shyamraj Syamalan	46	2	4.347826
Aditya Narse	93	2	2.150538
Jaswant Uppal	102	2	1.960784
Giri K	183	2	1.092896

1.1.9 9. Conclusion

While there are a lot of improvements that could be made from a data collection, accuracy, and normalization perspective, there are still a lot of interesting analyses that can be derived from the existing dataset. In this project, I have structured the dataset in a more organized way in order to conduct more accurate analysis by creating new columns and changing existing ones. I have analyzed the Core Integra Talent Acquisition Team Performance as well as the Client Fulfillment, and this analysis will be used to make a dashboard in order to better track KPIs for the team and set goals for better conversion rates and performance.