
sabrinasayed99 /
Fake-Job-Posts


Code
Issues
Pull requests
Actions
Projects
Wiki
Security


main
Fake-Job-Posts / Cleaning_NLP.ipynb


sabrinasayed99 XGBoost model
c1225e8 · 15 hours ago

7.88 MB

Cleaning the Data

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('/Users/sabrinasayed/Documents/GitHub/Fake-Job-Posts/Data
```

In [3]:
```
df.head()
```

Out[3]:

	job_id	title	location	department	salary_range	company_profile	
0	1	Marketing Intern	US, NY, New York	Marketing	NaN	We're Food52, and we've created a groundbreaki...	
1	2	Customer Service - Cloud Video Production	NZ, , Auckland	Success	NaN	90 Seconds, the worlds Cloud Video Production ...	(
2	3	Commissioning Machinery Assistant (CMA)	US, IA, Wever	NaN	NaN	Valor Services provides Workforce Solutions th...	
3	4	Account Executive - Washington DC	US, DC, Washington	Sales	NaN	Our passion for improving quality of life thro...	-
4	5	Bill Review Manager	US, FL, Fort Worth	NaN	NaN	SpotSource Solutions LLC is a Global Human Cap...	M

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17880 entries, 0 to 17879
Data columns (total 18 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   job_id              17880 non-null  int64 
 1   title               17880 non-null  object
```

```
 2   location             17534 non-null  object
 3   department           6333 non-null   object
 4   salary_range         2868 non-null   object
 5   company_profile      14572 non-null  object
 6   description          17879 non-null  object
 7   requirements         15184 non-null  object
 8   benefits             10668 non-null  object
 9   telecommuting        17880 non-null  int64
 10  has_company_logo     17880 non-null  int64
 11  has_questions        17880 non-null  int64
 12  employment_type      14409 non-null  object
 13  required_experience  10830 non-null  object
 14  required_education   9775 non-null   object
 15  industry             12977 non-null  object
 16  function             11425 non-null  object
 17  fraudulent           17880 non-null  int64
dtypes: int64(5), object(13)
memory usage: 2.5+ MB
```

In [5]:

```
# Dimensions
df.shape
```

Out[5]:

```
(17880, 18)
```

This data set has 17880 rows, 18 columns.

- job_id
- title
- location
- department
- salary_range
- company_profile
- description
- requirements
- benefit
- telecommuting
- has_company_logo
- has_question
- employment_type
- required_experience
- required_education
- industry
- function
- fraudulent

In [6]:

```
df['fraudulent'].value_counts()
```

Out[6]:

```
fraudulent
0    17014
1      866
Name: count, dtype: int64
```

There are 800 fake job posts in the dataset. Most of the data is catgorical/textual data.

In [7]:
```
# Remove leading and trailing spaces from column names
df.columns = df.columns.str.strip()
```

In [8]:
```
df.describe()
```

Out[8]:

	job_id	telecommuting	has_company_logo	has_questions	fraudule
count	17880.000000	17880.000000	17880.000000	17880.000000	17880.0000
mean	8940.500000	0.042897	0.795302	0.491723	0.0484
std	5161.655742	0.202631	0.403492	0.499945	0.2146
min	1.000000	0.000000	0.000000	0.000000	0.0000
25%	4470.750000	0.000000	1.000000	0.000000	0.0000
50%	8940.500000	0.000000	1.000000	0.000000	0.0000
75%	13410.250000	0.000000	1.000000	1.000000	0.0000
max	17880.000000	1.000000	1.000000	1.000000	1.0000

In [9]:
```
df.corr(numeric_only=True)
```

Out[9]:

	job_id	telecommuting	has_company_logo	has_questions	fr
job_id	1.000000	-0.004559	-0.014539	-0.087025	
telecommuting	-0.004559	1.000000	-0.019836	0.020345	
has_company_logo	-0.014539	-0.019836	1.000000	0.233932	
has_questions	-0.087025	0.020345	0.233932	1.000000	
fraudulent	0.079872	0.034523	-0.261971	-0.091627	

Dealing with missing values

In [10]:
```
# Check for missing values
df.isnull().sum()
```

Out[10]:
```
job_id                   0
title                    0
location               346
department           11547
salary_range         15012
company_profile       3308
description              1
requirements          2696
benefits              7212
```

```
telecommuting              0
has_company_logo           0
has_questions              0
employment_type         3471
required_experience     7050
required_education      8105
industry                4903
function                6455
fraudulent                 0
dtype: int64
```

In [11]:
```
# Percentage of missing values by column
df.isnull().sum() / df.shape[0] * 100
```

Out[11]:
```
job_id                  0.000000
title                   0.000000
location                1.935123
department             64.580537
salary_range           83.959732
company_profile        18.501119
description             0.005593
requirements           15.078300
benefits               40.335570
telecommuting           0.000000
has_company_logo        0.000000
has_questions           0.000000
employment_type        19.412752
required_experience    39.429530
required_education     45.329978
industry               27.421700
function               36.101790
fraudulent              0.000000
dtype: float64
```

In [12]:
```
# Drop location, job id, and salary range - they are either unnecessary o

df.drop(columns=['job_id', 'location', 'salary_range'], inplace=True)
```

Columns with highest percentage of missing values:

- Department
- Salary Range
- Benefits
- Required Education
- Required Experience

In [13]:
```
df.head()
```

Out[13]:

	title	department	company_profile	description	requir
0	Marketing Intern	Marketing	We're Food52, and we've created a groundbreaki...	Food52, a fast-growing, James Beard Award-winn...	Experie content mana syster

1	Customer Service - Cloud Video Production	Success	90 Seconds, the worlds Cloud Video Production ...	Organised - Focused - Vibrant - Awesome!Do you...	What we exp you:' respo
2	Commissioning Machinery Assistant (CMA)	NaN	Valor Services provides Workforce Solutions th...	Our client, located in Houston, is actively se...	Implem commissio commissi
3	Account Executive - Washington DC	Sales	Our passion for improving quality of life thro...	THE COMPANY: ESRI – Environmental Systems Rese...	EDUCATION: Ba or Master'
4	Bill Review Manager	NaN	SpotSource Solutions LLC is a Global Human Cap...	JOB TITLE: Itemization Review ManagerLOCATION:...	QUALIFICATI license in the

In [14]:
```
# Replace missing values in categorical columns with empty strings
columns_with_text = ['title', 'department', 'company_profile', 'descriptio
       'benefits', 'employment_type', 'required_experience',
       'required_education', 'industry', 'function']
df[columns_with_text] = df[columns_with_text].replace(np.nan, '')
df.isnull().sum()
```

Out[14]:
```
title                  0
department             0
company_profile        0
description            0
requirements           0
benefits               0
telecommuting          0
has_company_logo       0
has_questions          0
employment_type        0
required_experience    0
required_education     0
industry               0
function               0
fraudulent             0
dtype: int64
```

In [15]:
```
df.head()
```

Out[15]:

	title	department	company_profile	description	requir
0	Marketing Intern	Marketing	We're Food52, and we've created a groundbreaki...	Food52, a fast-growing, James Beard Award-winn...	Experie content mana syster

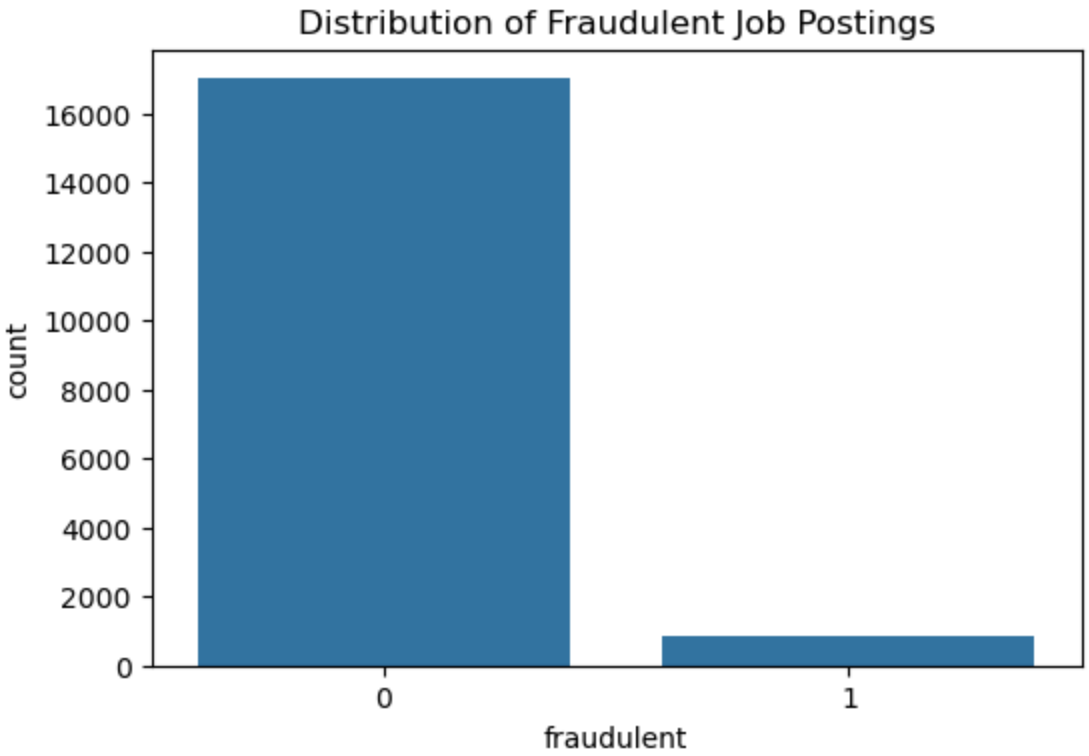
1	Customer Service - Cloud Video Production	Success	90 Seconds, the worlds Cloud Video Production ...	Organised - Focused - Vibrant - Awesome!Do you...	What we exp you:` respo
2	Commissioning Machinery Assistant (CMA)		Valor Services provides Workforce Solutions th...	Our client, located in Houston, is actively se...	Implem commissio commissi
3	Account Executive - Washington DC	Sales	Our passion for improving quality of life thro...	THE COMPANY: ESRI – Environmental Systems Rese...	EDUCATION: Ba or Master'
4	Bill Review Manager		SpotSource Solutions LLC is a Global Human Cap...	JOB TITLE: Itemization Review ManagerLOCATION:...	QUALIFICATI license in the

In [16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17880 entries, 0 to 17879
Data columns (total 15 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   title                17880 non-null  object
 1   department           17880 non-null  object
 2   company_profile      17880 non-null  object
 3   description          17880 non-null  object
 4   requirements         17880 non-null  object
 5   benefits             17880 non-null  object
 6   telecommuting        17880 non-null  int64 
 7   has_company_logo     17880 non-null  int64 
 8   has_questions        17880 non-null  int64 
 9   employment_type      17880 non-null  object
 10  required_experience  17880 non-null  object
 11  required_education   17880 non-null  object
 12  industry             17880 non-null  object
 13  function             17880 non-null  object
 14  fraudulent           17880 non-null  int64 
dtypes: int64(4), object(11)
memory usage: 2.0+ MB
```

Exploring Fradulent Job Postings

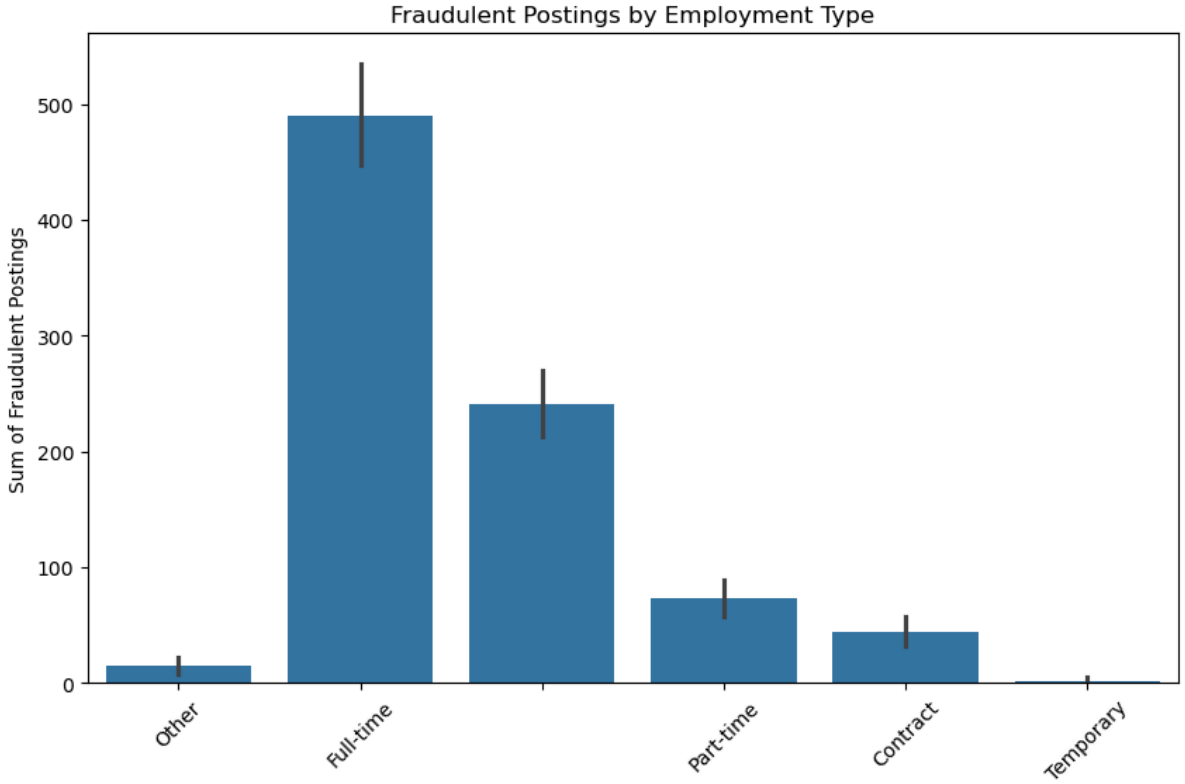
In [17]:
```
plt.figure(figsize=(6, 4))
sns.countplot(x='fraudulent', data=df)
plt.title('Distribution of Fraudulent Job Postings')
plt.show()
```



We'll be working with class imbalance

In [18]:

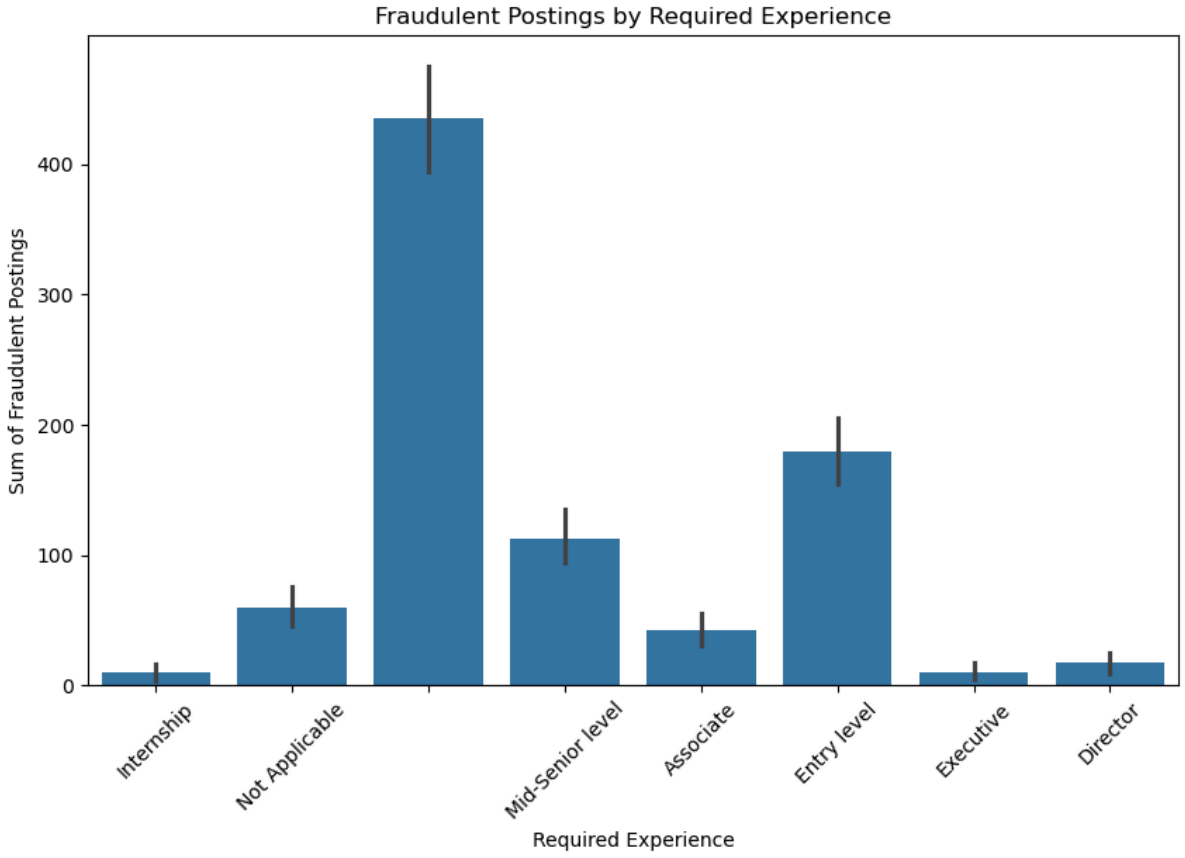
```
plt.figure(figsize=(10, 6))
sns.barplot(data=df, x='employment_type', y='fraudulent', estimator=sum)
plt.title('Fraudulent Postings by Employment Type')
plt.xlabel('Employment Type')
plt.ylabel('Sum of Fraudulent Postings')
plt.xticks(rotation=45)
plt.show()
```



Most fraudulent job postings are listed as full-time jobs.

In [19]:

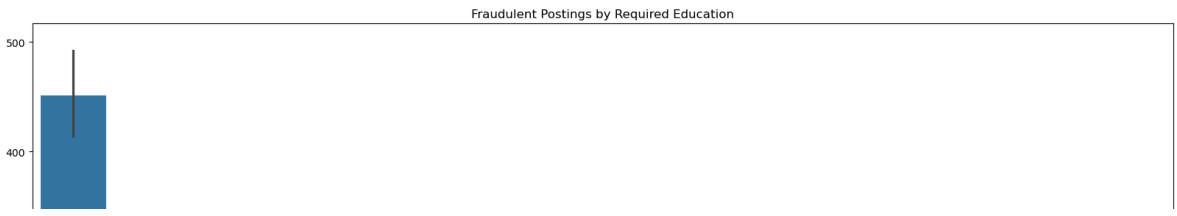
```
plt.figure(figsize=(10, 6))
sns.barplot(data=df, x='required_experience', y='fraudulent', estimator=su
plt.title('Fraudulent Postings by Required Experience')
plt.xlabel('Required Experience')
plt.ylabel('Sum of Fraudulent Postings')
plt.xticks(rotation=45)
plt.show()
```

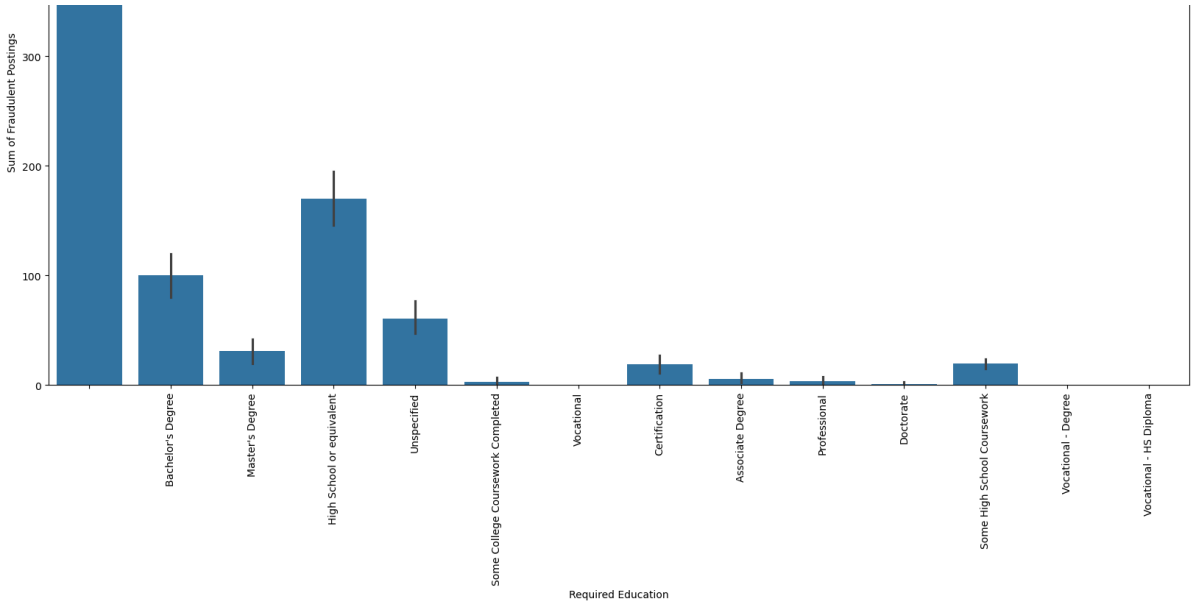


Most fraudulent job postings don't specify the required experience or target entry-level candidates.

In [20]:

```
plt.figure(figsize=(20, 10))
sns.barplot(data=df, x='required_education', y='fraudulent', estimator=sur
plt.title('Fraudulent Postings by Required Education')
plt.xlabel('Required Education')
plt.ylabel('Sum of Fraudulent Postings')
plt.xticks(rotation = 90)
plt.show()
```



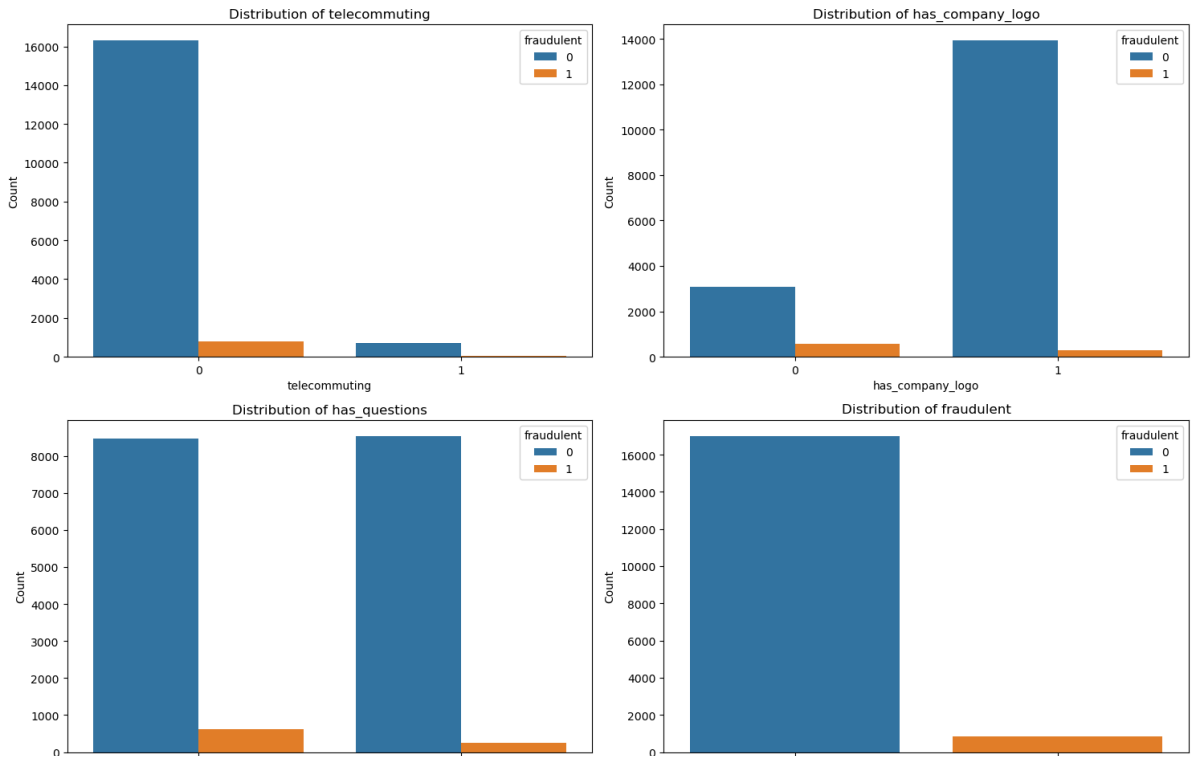


Most fraudulent job postings don't specify the required education or target candidates with a high school diploma or GED.

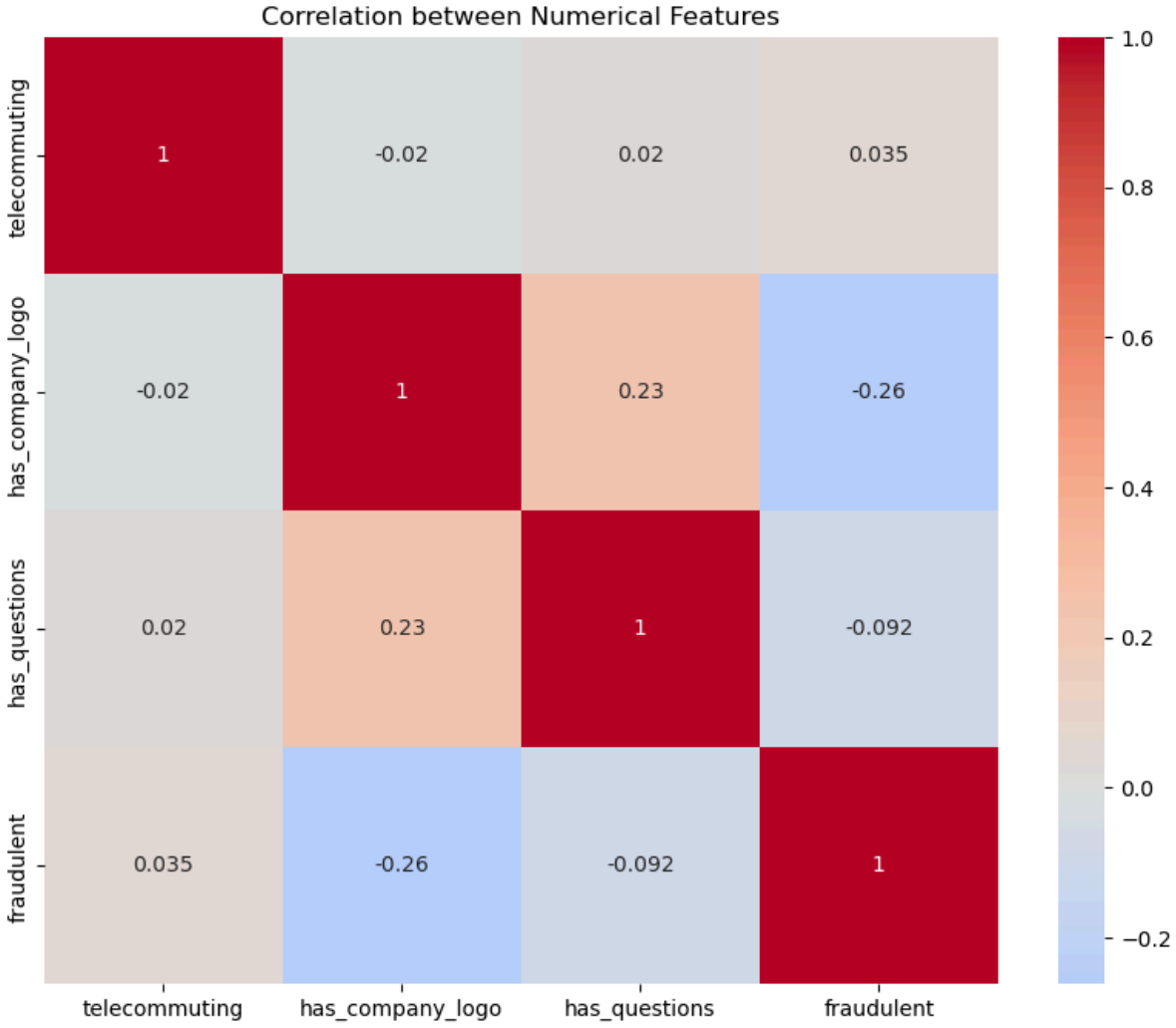
In [58]:

```
numerical_cols = ['telecommuting', 'has_company_logo', 'has_questions', '

# Explore distributions
plt.figure(figsize=(15, 10))
for i, col in enumerate(numerical_cols, 1):
    plt.subplot(2, 2, i)
    sns.countplot(data=df, x=col)
    plt.title(f'Distribution of {col}')
    plt.xlabel(col)
    plt.ylabel('Count')
plt.tight_layout()
plt.show()
```



In [59]:
```
# Correlation Heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(df[numerical_cols].corr(),
            annot=True,
            cmap='coolwarm',
            center=0)
plt.title('Correlation between Numerical Features')
plt.show()
```



Low correlations between the numerical features and target

In [21]:
```
# Explore what fraudulent job postings descriptions look like
df[df['fraudulent'] == True]['description'].iloc[0]
```

Out[21]:
```
'IC&amp;amp;E Technician | Bakersfield, CA Mt. PosoPrincipal Duties and Respo
nsibilities:\xa0Calibrates, tests, maintains, troubleshoots, and installs
all power plant instrumentation, control systems and electrical equipmen
t.Performs maintenance on motor control centers, motor operated valves, g
enerators, excitation equipment and motors.Performs preventive, predictiv
e and corrective maintenance on equipment, coordinating work with various
team members.Designs and installs new equipment and/or system modificatio
ns.Troubleshoots and performs maintenance on DC backup power equipment, p
rocess controls, programmable logic controls (PLC), and emission monitori
```

```
process controls, programmable logic controls (PLC), and emission monitori
ng equipment.Uses maintenance reporting system to record time and materia
l use, problem identified and corrected, and further action required; pro
vides complete history of maintenance on equipment.Schedule, coordinate,
work with and monitor contractors on specific tasks, as required.Follows
safe working practices at all times.Identifies safety hazards and recomme
nds solutions.Follows environmental compliance work practices.Identifies
environmental non-compliance problems and assist in implementing solution
s.Assists other team members and works with all departments to support ge
nerating station in achieving their performance goals.Trains other team m
embers in the areas of instrumentation, control, and electrical systems.P
erforms housekeeping assignments, as directed.Conduct equipment and syste
m tagging according to company and plant rules and regulations.Perform eq
uipment safety inspections, as required, and record results as appropriat
e.\xa0Participate in small construction projects.\xa0 Read and interpret
drawings, sketches, prints, and specifications, as required.Orders parts
as needed to affect maintenance and repair.Performs Operations tasks on a
n as-needed basis and other tasks as assigned.Available within a reasonab
le response time for emergency call-ins and overtime, plus provide accept
able off-hour contact by phone and company pager.\xa0\xa0\xa0\xa0\xa0\xa0
\xa0\xa0\xa0 Excellent Verbal and Written Communications Skills:Ability t
o coordinate work activities with other team members on technical subject
s across job families.Ability to work weekends, holidays, and rotating sh
ifts, as required.'
```

In [22]:
```
df[df['fraudulent'] == True]['description'].iloc[100]
```

Out[22]:
```
'We are a full-service marketing and staffing firm, serving companies ran
ging from Fortune 100 to new start-up organizations. We work with job see
kers in an equally broad range, from light industrial temporary workers t
o executive level candidates.Are you looking for a Work from Home Opportu
nity where you can earn up to $2500 and more per week? Our Online Service
Representative position would be perfect for you!- Set your own hours - M
ake money every time you decide to work - Work remotely from home - Get p
aid weekly - If you have a computer with internet, this is for you'
```

Cleaning the Data - Text Preprocessing

In [23]:
```
import re
import nltk

from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
nltk.download('omw-1.4')

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt_tab')
```

```
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     /Users/sabrinasayed/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/sabrinasayed/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/sabrinasayed/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/sabrinasayed/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt_tab to
[nltk_data]     /Users/sabrinasayed/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
```

Out[23]: True

In [24]:
```python
def preprocess_text(text):
    if not isinstance(text, str):
        return ''

    # Convert to lowercase
    text = text.lower()

    # Remove HTML tags
    text = re.sub(r'<.*?>', '', text)

    # Remove special characters and digits
    text = re.sub(r'[^\w\s]', ' ', text)
    text = re.sub(r'\d+', '', text)

    # Remove extra whitespace
    text = ' '.join(text.split())

    # Tokenization
    tokens = word_tokenize(text)

    # Remove stopwords
    stop_words = set(stopwords.words('english'))
    tokens = [token for token in tokens if token not in stop_words]

    # Lemmatization
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(token) for token in tokens]

    # Join tokens back into text
    return ' '.join(tokens)
```

In [25]:
```python
# Apply preprocessing to relevant text columns
text_columns = ['title', 'company_profile', 'description', 'requirements',
for column in text_columns:
    df[f'{column}_processed'] = df[column].apply(preprocess_text)

df.head()
```

Out[25]:

	title	department	company_profile	description	requir
0	Marketing Intern	Marketing	We're Food52, and we've created a groundbreaki...	Food52, a fast-growing, James Beard Award-winn...	Experie content mana syster
	Customer		90 Seconds, the		

1	Customer Service - Cloud Video Production	Success	90 Seconds, the worlds Cloud Video Production ...	Organised - Focused - Vibrant - Awesome!Do you...	What we exp you:` respo
2	Commissioning Machinery Assistant (CMA)		Valor Services provides Workforce Solutions th...	Our client, located in Houston, is actively se...	Implem commissio commissi
3	Account Executive - Washington DC	Sales	Our passion for improving quality of life thro...	THE COMPANY: ESRI – Environmental Systems Rese...	EDUCATION: Ba or Master'
4	Bill Review Manager		SpotSource Solutions LLC is a Global Human Cap...	JOB TITLE: Itemization Review ManagerLOCATION:...	QUALIFICATI license in the

In [26]:
```
df['description_processed'].iloc[0]
```

Out[26]:
```
'food fast growing james beard award winning online food community crowd
sourced curated recipe hub currently interviewing full part time unpaid i
ntern work small team editor executive developer new york city headquarte
rs reproducing repackaging existing food content number partner site huff
ington post yahoo buzzfeed various content management systemsresearching
blog website provision food affiliate programassisting day day affiliate
program support screening affiliate assisting affiliate inquiriessupporti
ng pr amp event neededhelping office administrative work filing mailing p
reparing meetingsworking developer document bug suggest improvement sites
upporting marketing executive staff'
```

In [27]:
```
df['requirements_processed'].iloc[10]
```

Out[27]:
```
'position url_fdaaebceedefdcefecbafedd developerjob location united state
new jersey jersey cityus work status required ead green card u citizen de
tailed description url_fdaaebceedefdcefecbafedd developer strong sql amp
vb net working highly effective software development team responsible dev
elopment enhancement idb reporting billing system work effort performed a
ccording policy procedure relating software development quality backgroun
d developer year development experience using url_fdaaebceedefdcefecbafed
d vb net including year sql development required college degree experienc
e building web based application using url_fdaaebceedefdcefecbafedd sql s
erver expert level hand experience writing sql server stored procedure pr
oficient m office product excel access word outlook worked role involved
creation report internal management external customer experience developi
ng another scripting front end language excellent communication skillsif
interested please send updated profile email_dfbffcfbedeaaedaccfeee hirin
g sale account director advertising digital sale position sale account di
rector pacific northwestjob location united state washington seattleus wo
rk status required ead green card u citizen requirement year advertising
digital sale experience preferrably region ability provide forecasting in
formation management proven success prospecting cold calling regional acc
ount proven experience automotive tech health care travel category among
```

```
others specialized experience hand agency account management self starter
roll sleeve mentality ability work internal team solve problem solid gras
p interactive medium including various pricing model targeting technology
ad serving bachelor degree strong long distance communication skill using
email instant messenger platform phone high technical aptitude learning w
orking within desktop web based application window microsoft office dsps
ad exchange ability calculate analyze data based standard digital adverti
sing cost per metric cpm cpc cpa etc detail oriented strong organizationa
l skill multitasking ability ability work efficiently effectively tight d
eadline strong sense urgency personality work effectively within fast mov
ing environment many different type people desire ability work home'
```

Exploring Textual data

In [28]:

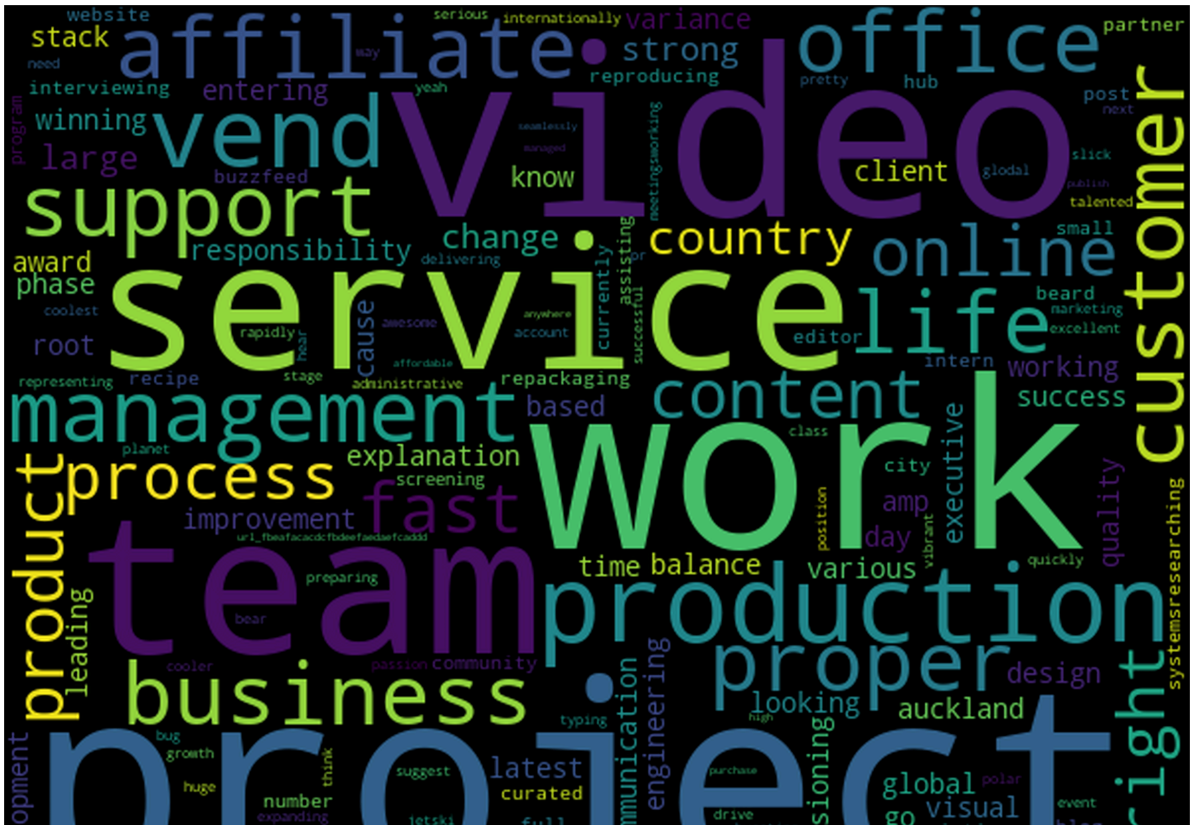
```
from nltk import ngrams
from wordcloud import WordCloud, STOPWORDS

def generate(string,ngram):
    n_grams=ngrams(word_tokenize(string),ngram)
    grams=[" ".join(val) for val in n_grams]
    return grams

real = df[df['fraudulent']==0]['description_processed'].values
cloud = WordCloud(width= 600, height= 600, stopwords= STOPWORDS,
                      background_color='black').generate(str(real))

fig = plt.figure(figsize = (30, 30))
plt.imshow(cloud, interpolation= 'blackman')
plt.axis('off')
```

Out[28]: (-0.5, 599.5, 599.5, -0.5)



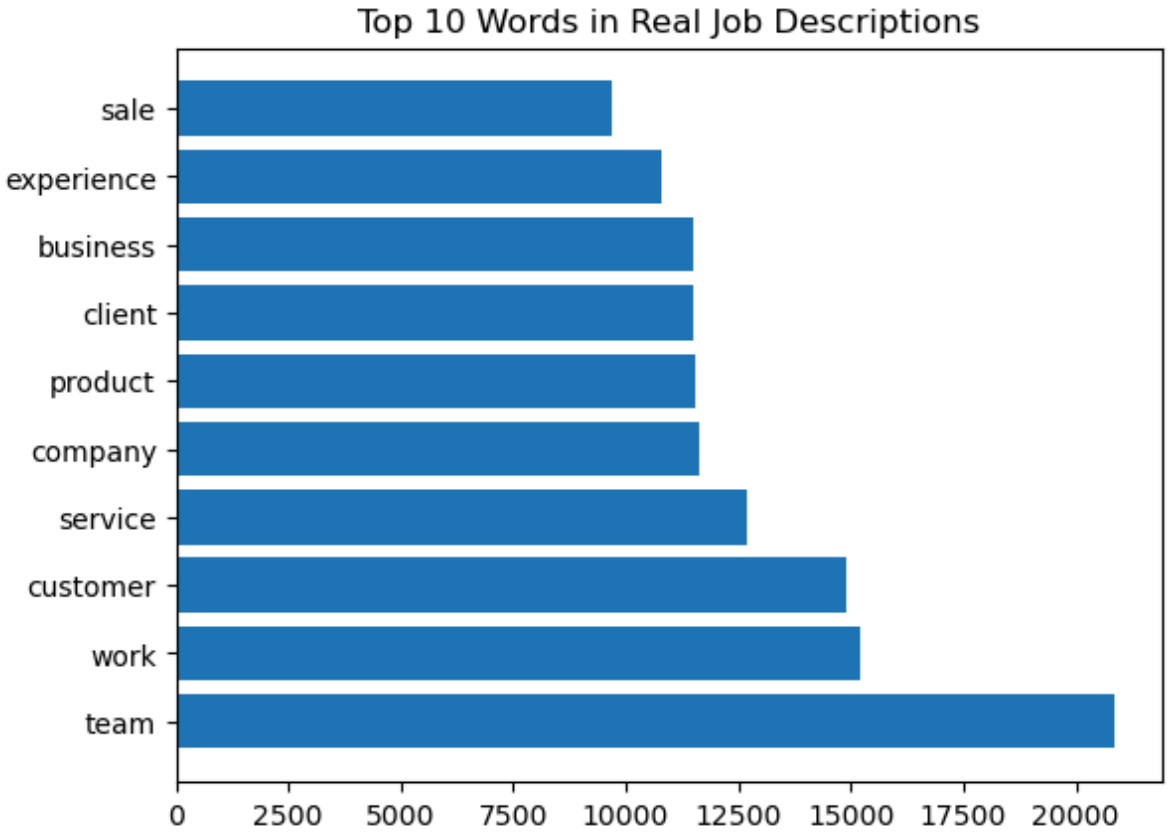
In [29]:

```
from collections import defaultdict
dict = defaultdict(int)

for text in df[df["fraudulent"]==0]["description_processed"]:
    for words in generate(text,1):
        dict[words]+=1

word_freq=pd.DataFrame(sorted(dict.items(),key=lambda x: x[1],reverse=True
plt.barh(word_freq[0][:10], word_freq[1][:10])
plt.title('Top 10 Words in Real Job Descriptions')
```

Out[29]: Text(0.5, 1.0, 'Top 10 Words in Real Job Descriptions')



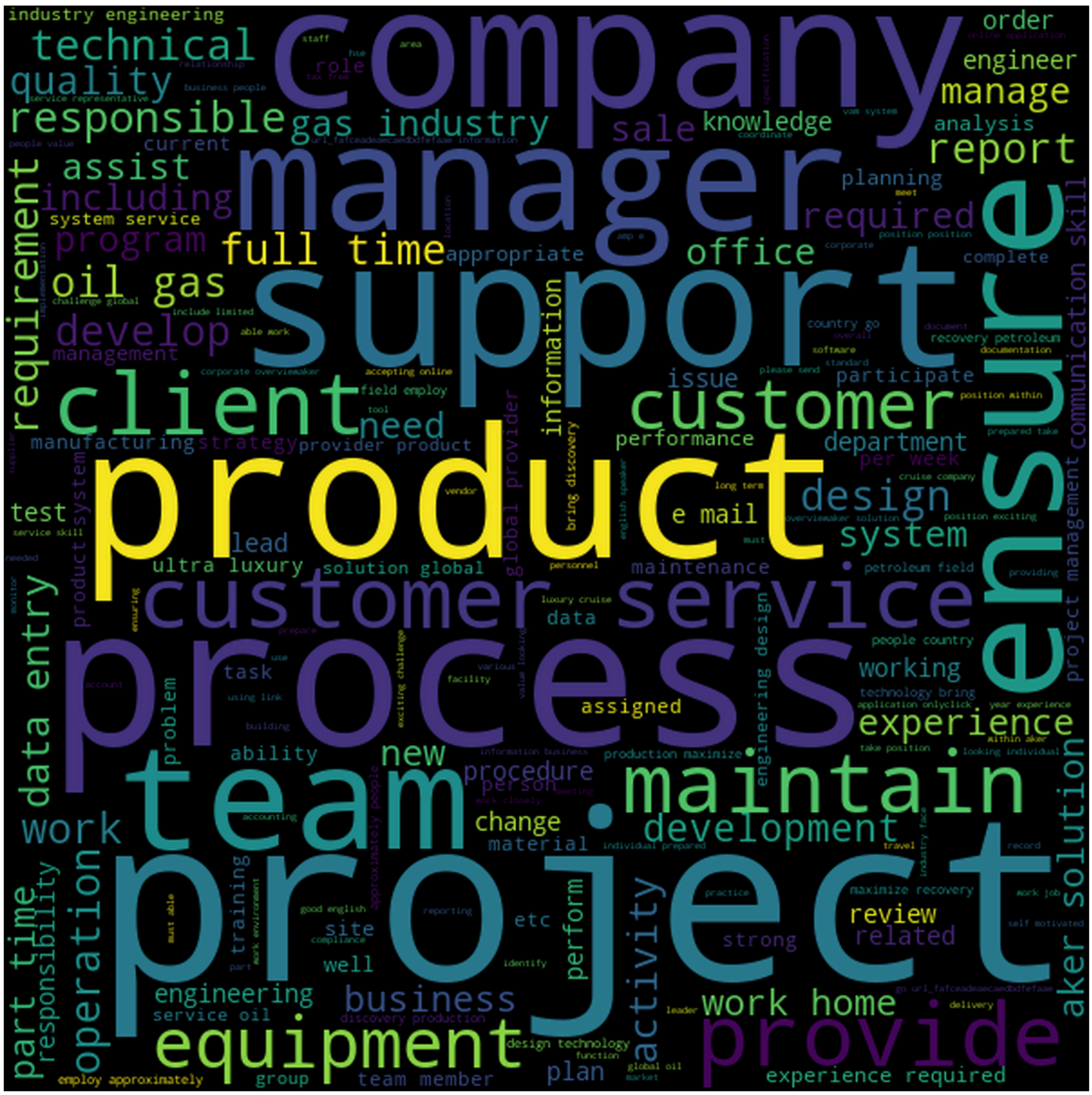
In [30]:

```
# Generate word cloud for fake job postings
fake = df[df['fraudulent']==1]['description_processed'].values
cloud = WordCloud(width= 600, height= 600, stopwords= STOPWORDS,
                  background_color='black').generate(str(fake))

fig = plt.figure(figsize = (30, 30))
```

```
plt.imshow(cloud, interpolation='blackman')
plt.axis('off')
```

Out[30]: (-0.5, 599.5, 599.5, -0.5)



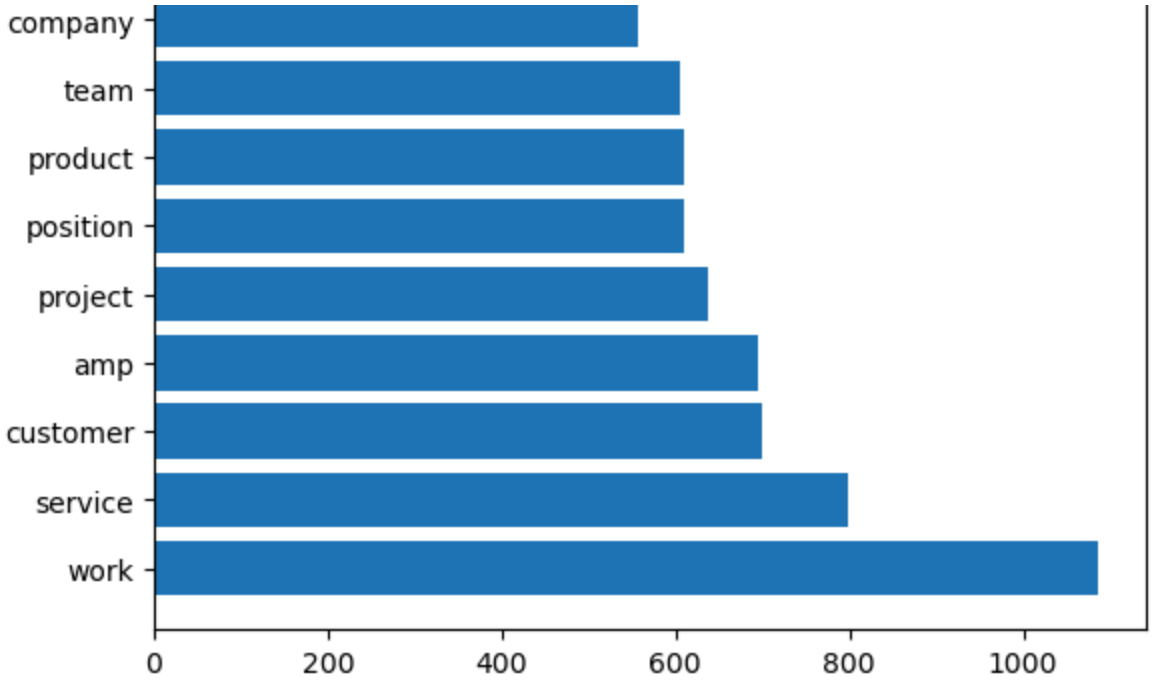
In [31]:

```
dict_2 = defaultdict(int)

for text in df[df["fraudulent"]==1]["description_processed"]:
    for words in generate(text,1):
        dict_2[words]+=1

word_freq_2=pd.DataFrame(sorted(dict_2.items(),key=lambda x: x[1],reverse=
plt.barh(word_freq_2[0][:10], word_freq_2[1][:10])
plt.title('Top 10 Words in Fake Job Descriptions')
```

Out[31]: Text(0.5, 1.0, 'Top 10 Words in Fake Job Descriptions')



In [32]:
```
df.head()
```

Out[32]:

	title	department	company_profile	description	requir
0	Marketing Intern	Marketing	We're Food52, and we've created a groundbreaki...	Food52, a fast-growing, James Beard Award-winn...	Experie content mana syster
1	Customer Service - Cloud Video Production	Success	90 Seconds, the worlds Cloud Video Production ...	Organised - Focused - Vibrant - Awesome!Do you...	What we exp you: respo
2	Commissioning Machinery Assistant (CMA)		Valor Services provides Workforce Solutions th...	Our client, located in Houston, is actively se...	Implem commissio commissi
3	Account Executive - Washington DC	Sales	Our passion for improving quality of life thro...	THE COMPANY: ESRI – Environmental Systems Rese...	EDUCATION: Ba or Master'
4	Bill Review Manager		SpotSource Solutions LLC is a Global Human Cap...	JOB TITLE: Itemization Review ManagerLOCATION:...	QUALIFICATI license in the

In [33]:
```
df.info()
```

<class 'pandas.core.frame.DataFrame'>

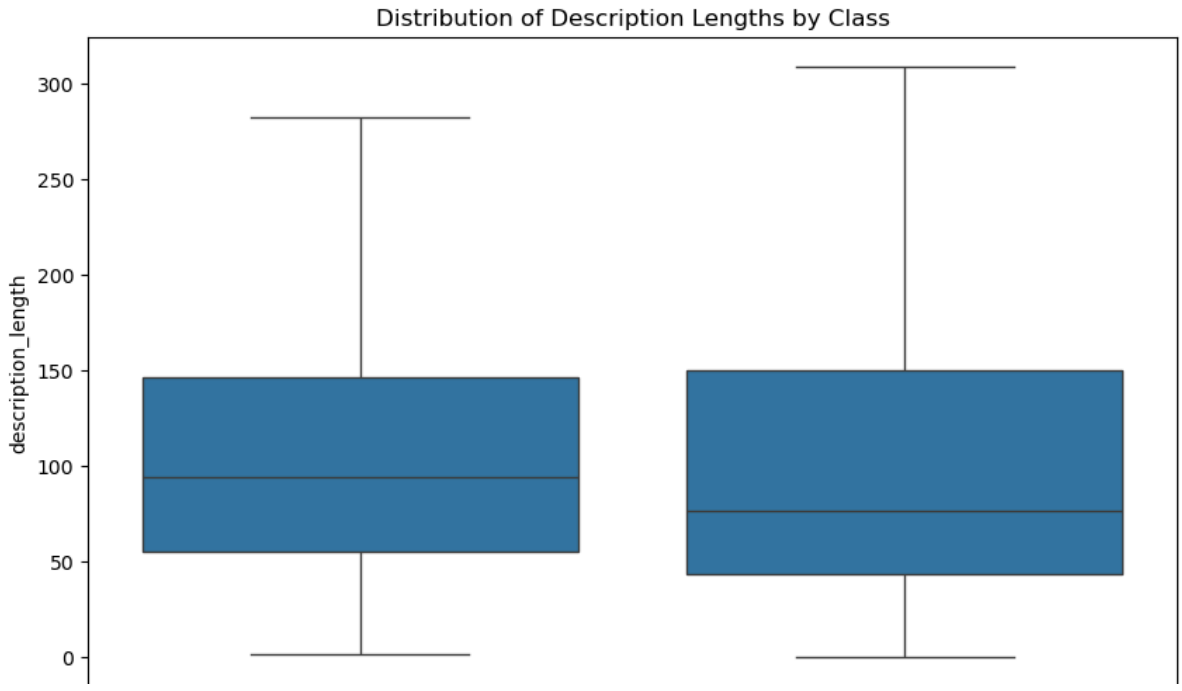
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17880 entries, 0 to 17879
Data columns (total 20 columns):
 #   Column                     Non-Null Count  Dtype 
---  ------                     --------------  ----- 
 0   title                      17880 non-null  object
 1   department                 17880 non-null  object
 2   company_profile            17880 non-null  object
 3   description                17880 non-null  object
 4   requirements               17880 non-null  object
 5   benefits                   17880 non-null  object
 6   telecommuting              17880 non-null  int64 
 7   has_company_logo           17880 non-null  int64 
 8   has_questions              17880 non-null  int64 
 9   employment_type            17880 non-null  object
 10  required_experience        17880 non-null  object
 11  required_education         17880 non-null  object
 12  industry                   17880 non-null  object
 13  function                   17880 non-null  object
 14  fraudulent                 17880 non-null  int64 
 15  title_processed            17880 non-null  object
 16  company_profile_processed  17880 non-null  object
 17  description_processed      17880 non-null  object
 18  requirements_processed     17880 non-null  object
 19  benefits_processed         17880 non-null  object
dtypes: int64(4), object(16)
memory usage: 2.7+ MB
```

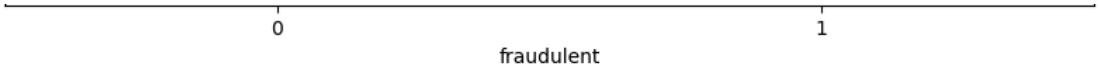
In [61]:

```
# Analyze text lengths
df['description_length'] = df['description_processed'].str.split().str.le

plt.figure(figsize=(10, 6))
sns.boxplot(x='fraudulent', y='description_length', data=df,showfliers=Fa
plt.title('Distribution of Description Lengths by Class')
plt.show()

# Basic statistics
print(df.groupby('fraudulent')['description_length'].describe())
```





fraudulent	count	mean	std	min	25%	50%	75%	max
0	17014.0	110.989538	79.442735	1.0	55.0	94.0	146.0	1419.0
1	866.0	107.105081	92.603321	0.0	43.0	76.0	150.0	806.0

Getting most common words by class

In [36]:
```
import re
from collections import Counter

def check_special_chars(text_series):
    # Join all text
    all_text = ' '.join(text_series)
    # Find all non-alphabetic characters
    special_chars = re.findall(r'[^a-zA-Z\s]', all_text)
    return Counter(special_chars).most_common()

print("Special characters remaining in processed text:")
print(check_special_chars(df['description_processed']))
```

```
Special characters remaining in processed text:
[('_', 8945), ('α', 2876), ('ε', 2153), ('ι', 2022), ('τ', 1997), ('ο', 188
6), ('ν', 1623), ('σ', 1277), ('ρ', 1145), ('μ', 1077), ('π', 1011), ('η',
1008), ('κ', 968), ('ς', 952), ('υ', 949), ('ί', 630), ('λ', 628), ('γ', 62
4), ('έ', 544), ('ά', 531), ('ό', 447), ('δ', 441), ('χ', 365), ('ή', 329),
('ω', 297), ('θ', 266), ('ύ', 235), ('ώ', 222), ('φ', 209), ('ζ', 125),
('β', 110), ('ξ', 105), ('é', 70), ('로', 61), ('ร', 54), ('고', 52), ('을',
48), ('น', 46), ('ś', 45), ('는', 45), ('니', 44), ('ㄱ', 44), ('이', 43),
('บ', 41), ('서', 40), ('다', 40), ('하', 39), ('å', 39), ('스', 39), ('ง', 3
9), ('ψ', 38), ('ł', 37), ('있', 36), ('일', 33), ('으', 33), ('ย', 33),
('에', 31), ('ม', 31), ('한', 30), ('ć', 30), ('무', 30), ('ก', 30), ('화', 2
8), ('ę', 28), ('ó', 27), ('버', 27), ('를', 27), ('의', 27), ('ø', 27),
('ï', 25), ('ą', 25), ('업', 25), ('ı', 25), ('ü', 24), ('즈', 24), ('인', 2
3), ('바', 23), ('â', 23), ('빌', 23), ('ä', 22), ('리', 22), ('면', 22),
('가', 22), ('ล', 22), ('글', 21), ('벌', 21), ('적', 21), ('모', 21), ('자',
21), ('ー', 21), ('的', 21), ('ะ', 21), ('관', 20), ('비', 19), ('해', 19),
('재', 18), ('과', 17), ('시', 17), ('본', 16), ('광', 16), ('발', 16), ('습',
16), ('저', 16), ('드', 16), ('ว', 16), ('ป', 16), ('국', 15), ('트', 15),
('業', 15), ('แ', 15), ('ด', 15), ('ย', 15), ('ใ', 15), ('수', 14), ('개', 1
4), ('ë', 14), ('告', 14), ('ข', 14), ('ท', 14), ('ต', 14), ('지', 13),
('정', 13), ('사', 13), ('은', 13), ('기', 13), ('ห', 13), ('ค', 13), ('만', 1
2), ('및', 12), ('ż', 12), ('도', 12), ('주', 12), ('の', 12), ('성', 11),
('신', 11), ('ï', 11), ('첫', 11), ('할', 11), ('오', 11), ('세', 11), ('요',
11), ('디', 11), ('대', 10), ('라', 10), ('크', 10), ('보', 10), ('내', 10),
('を', 10), ('ン', 10), ('在', 10), ('등', 9), ('ö', 9), ('공', 9), ('피', 9),
('매', 9), ('게', 9), ('バ', 9), ('ト', 9), ('理', 9), ('以', 9), ('ษ', 9),
('현', 8), ('금', 8), ('란', 8), ('직', 8), ('며', 8), ('위', 8), ('외', 8),
('ル', 8), ('広', 8), ('て', 8), ('本', 8), ('及', 8), ('š', 8), ('경', 8),
('ช', 8), ('ส', 8), ('분', 7), ('잠', 7), ('명', 7), ('확', 7), ('와', 7),
('계', 7), ('나', 7), ('당', 7), ('될', 7), ('어', 7), ('è', 7), ('영', 7),
('日', 7), ('し', 7), ('管', 7), ('提', 7), ('作', 7), ('與', 7), ('用', 7),
('ไ', 7), ('จ', 7), ('마', 6), ('팅', 6), ('열', 6), ('용', 6), ('변', 6),
('역', 6), ('입', 6), ('두', 6), ('유', 6), ('허', 6), ('린', 6), ('출', 6),
('들', 6), ('쾅', 6), ('は', 6), ('モ', 6), ('イ', 6), ('に', 6), ('い', 6),
('る', 6), ('営', 6), ('と', 6), ('リ', 6), ('内', 6), ('務', 6), ('ń', 6),
('創', 6), ('工', 6), ('使', 6), ('廣', 6), ('有', 6), ('市', 6), ('行', 6),
('ณ', 6), ('พ', 6), ('파', 5), ('너', 5), ('간', 5), ('키', 5), ('원', 5),
```

```
('장', 5), ('타', 5), ('꿈', 5), ('상', 5), ('합', 5), ('격', 5), ('담', 5),
('구', 5), ('안', 5), ('문', 5), ('ß', 5), ('受', 5), ('ス', 5), ('ク', 5),
('제', 5), ('者', 5), ('上', 5), ('회', 5), ('¼', 5), ('活', 5), ('我', 5),
('們', 5), ('並', 5), ('場', 5), ('台', 5), ('전', 4), ('립', 4), ('행', 4),
('력', 4), ('진', 4), ('fi', 4), ('찾', 4), ('플', 4), ('랫', 4), ('폼', 4),
('활', 4), ('혁', 4), ('프', 4), ('킬', 4), ('미', 4), ('앱', 4), ('중', 4),
('되', 4), ('었', 4), ('동', 4), ('아', 4), ('극', 4), ('반', 4), ('애', 4),
('네', 4), ('워', 4), ('ü', 4), ('æ', 4), ('ビ', 4), ('で', 4), ('事', 4),
('び', 4), ('ま', 4), ('す', 4), ('さ', 4), ('国', 4), ('韓', 4), ('ド', 4),
('컨', 4), ('텐', 4), ('츠', 4), ('ネ', 4), ('í', 4), ('환', 4), ('公', 4),
('司', 4), ('國', 4), ('成', 4), ('最', 4), ('佳', 4), ('生', 4), ('供', 4),
('訊', 4), ('動', 4), ('灣', 4), ('溝', 4), ('通', 4), ('험', 4), ('케', 3),
('법', 3), ('협', 3), ('통', 3), ('á', 3), ('fl', 3), ('근', 3), ('쿄', 3),
('설', 3), ('능', 3), ('더', 3), ('록', 3), ('름', 3), ('연', 3), ('희', 3),
('려', 3), ('ê', 3), ('ズ', 3), ('開', 3), ('発', 3), ('け', 3), ('ッ', 3),
('フ', 3), ('り', 3), ('ャ', 3), ('コ', 3), ('ア', 3), ('資', 3), ('휴', 3),
('것', 3), ('容', 3), ('携', 3), ('パ', 3), ('ナ', 3), ('운', 3), ('생', 3),
('셋', 3), ('新', 3), ('程', 3), ('所', 3), ('更', 3), ('息', 3), ('果', 3),
('分', 3), ('如', 3), ('您', 3), ('相', 3), ('關', 3), ('產', 3), ('將', 3),
('進', 3), ('之', 3), ('ฒ', 3), ('ๆ', 3), ('ฝ', 3), ('ᄀ', 3), ('캠', 2),
('페', 2), ('채', 2), ('널', 2), ('석', 2), ('최', 2), ('산', 2), ('굴', 2),
('ã', 2), ('½', 2), ('’', 2), ('심', 2), ('샌', 2), ('코', 2), ('년', 2),
('표', 2), ('불', 2), ('큰', 2), ('꿔', 2), ('야', 2), ('념', 2), ('날', 2),
('션', 2), ('달', 2), ('노', 2), ('락', 2), ('조', 2), ('체', 2), ('런', 2),
('탕', 2), ('남', 2), ('색', 2), ('향', 2), ('웹', 2), ('부', 2), ('받', 2),
('템', 2), ('축', 2), ('클', 2), ('언', 2), ('써', 2), ('루', 2), ('손', 2),
('번', 2), ('싶', 2), ('熱', 2), ('職', 2), ('め', 2), ('方', 2), ('待', 2),
('ち', 2), ('せ', 2), ('定', 2), ('年', 2), ('べ', 2), ('チ', 2), ('た', 2),
('그', 2), ('람', 2), ('主', 2), ('な', 2), ('代', 2), ('ワ', 2), ('テ', 2),
('ツ', 2), ('ジ', 2), ('興', 2), ('運', 2), ('à', 2), ('넘', 2), ('른', 2),
('누', 2), ('꼼', 2), ('감', 2), ('각', 2), ('월', 2), ('一', 2), ('企', 2),
('家', 2), ('共', 2), ('立', 2), ('為', 2), ('希', 2), ('望', 2), ('鎖', 2),
('屏', 2), ('讓', 2), ('研', 2), ('式', 2), ('重', 2), ('點', 2), ('能', 2),
('目', 2), ('前', 2), ('到', 2), ('了', 2), ('名', 2), ('部', 2), ('等', 2),
('地', 2), ('內', 2), ('找', 2), ('想', 2), ('長', 2), ('意', 2), ('是', 2),
('選', 2), ('協', 2), ('助', 2), ('規', 2), ('劃', 2), ('態', 2), ('播', 2),
('網', 2), ('服', 2), ('析', 2), ('人', 2), ('夥', 2), ('伴', 2), ('求', 2),
('例', 2), ('客', 2), ('戶', 2), ('品', 2), ('決', 2), ('團', 2), ('隊', 2),
('ฬ', 2), ('ก', 2), ('덕', 2), ('략', 1), ('집', 1), ('럴', 1), ('여', 1),
('효', 1), ('율', 1), ('예', 1), ('배', 1), ('규', 1), ('십', 1), ('필', 1),
('골', 1), ('쉬', 1), ('민', 1), ('착', 1), ('情', 1), ('持', 1), ('っ', 1),
('勤', 1), ('探', 1), ('プ', 1), ('ラ', 1), ('ォ', 1), ('ム', 1), ('ァ', 1),
('革', 1), ('命', 1), ('変', 1), ('化', 1), ('ハ', 1), ('ニ', 1), ('サ', 1),
('着', 1), ('役', 1), ('割', 1), ('お', 1), ('ソ', 1), ('か', 1), ('ら', 1),
('投', 1), ('총', 1), ('괄', 1), ('창', 1), ('든', 1), ('실', 1), ('질', 1),
('익', 1), ('했', 1), ('길', 1), ('우', 1), ('랍', 1), ('약', 1), ('충', 1),
('룰', 1), ('없', 1), ('함', 1), ('께', 1), ('딱', 1), ('맞', 1), ('担', 1),
('当', 1), ('れ', 1), ('店', 1), ('ど', 1), ('売', 1), ('げ', 1), ('全', 1),
('味', 1), ('が', 1), ('あ', 1), ('向', 1), ('思', 1), ('う', 1), ('ぜ', 1),
('ひ', 1), ('エ', 1), ('く', 1), ('だ', 1), ('급', 1), ('형', 1), ('マ', 1),
('界', 1), ('関', 1), ('係', 1), ('‘', 1), ('î', 1), ('량', 1), ('작', 1),
('따', 1), ('될', 1), ('베', 1), ('욱', 1), ('히', 1), ('완', 1), ('태', 1),
('좋', 1), ('낸', 1), ('결', 1), ('단', 1), ('흐', 1), ('징', 1), ('품', 1),
('복', 1), ('후', 1), ('簡', 1), ('介', 1), ('由', 1), ('群', 1), ('師', 1),
('同', 1), ('宗', 1), ('旨', 1), ('不', 1), ('斷', 1), ('改', 1), ('善', 1),
('透', 1), ('過', 1), ('螢', 1), ('幕', 1), ('當', 1), ('智', 1), ('慧', 1),
('手', 1), ('機', 1), ('渠', 1), ('道', 1), ('體', 1), ('驗', 1), ('美', 1),
('好', 1), ('發', 1), ('應', 1), ('豐', 1), ('富', 1), ('消', 1), ('多', 1),
('元', 1), ('努', 1), ('力', 1), ('享', 1), ('便', 1), ('利', 1), ('愉', 1),
('快', 1), ('卓', 1), ('越', 1), ('吸', 1), ('睛', 1), ('效', 1), ('止', 1),
```

```
('已', 1), ('禺', 1), ('後', 1), ('亦', 1), ('刻', 1), ('遲', 1), ('弟', 1),
('佔', 1), ('率', 1), ('約', 1), ('員', 1), ('總', 1), ('位', 1), ('於', 1),
('首', 1), ('爾', 1), ('東', 1), ('京', 1), ('舊', 1), ('金', 1), ('山', 1),
('和', 1), ('臺', 1), ('北', 1), ('設', 1), ('尋', 1), ('極', 1), ('速', 1),
('裡', 1), ('茁', 1), ('壯', 1), ('且', 1), ('對', 1), ('高', 1), ('度', 1),
('趣', 1), ('忱', 1), ('你', 1), ('擇', 1), ('營', 1), ('包', 1), ('括', 1),
('各', 1), ('種', 1), ('處', 1), ('聯', 1), ('項', 1), ('負', 1), ('責', 1),
('下', 1), ('驚', 1), ('績', 1), ('衷', 1), ('心', 1), ('功', 1), ('願', 1),
('接', 1), ('挑', 1), ('戰', 1), ('擁', 1), ('精', 1), ('神', 1), ('追', 1),
('遠', 1), ('大', 1), ('夢', 1), ('此', 1), ('編', 1), ('輯', 1), ('顯', 1),
('示', 1), ('連', 1), ('合', 1), ('表', 1), ('社', 1), ('交', 1), ('平', 1),
('落', 1), ('格', 1), ('聆', 1), ('聽', 1), ('需', 1), ('解', 1), ('問', 1),
('題', 1), ('究', 1), ('策', 1), ('擔', 1), ('任', 1), ('跨', 1), ('橋', 1),
('樑', 1), ('ศ', 1), ('ภ', 1), ('ฏ', 1), ('พ', 1), ('ไ', 1), ('ú', 1)]
```

There are still a lot of special characters remaining, a lot of which are non-english characters.

In [37]:

```python
import unicodedata

def preprocess_text(text):
    if not isinstance(text, str):
        return ''

    # Convert to lowercase
    text = text.lower()

    # Normalize unicode characters
    text = unicodedata.normalize('NFKD', text)

    # Remove non-ASCII characters
    text = text.encode('ascii', 'ignore').decode('ascii')

    # Remove HTML tags
    text = re.sub(r'<.*?>', '', text)

    # Remove special characters and digits
    text = re.sub(r'[^\w\s]', ' ', text)
    text = re.sub(r'\d+', '', text)

    # Remove extra whitespace
    text = ' '.join(text.split())

    # Tokenization
    tokens = word_tokenize(text)

    # Remove stopwords
    stop_words = set(stopwords.words('english'))
    tokens = [token for token in tokens if token not in stop_words]

    # Lemmatization
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(token) for token in tokens]

    # Join tokens back into text
    return ' '.join(tokens)

# Reapply the improved preprocessing
text_columns = ['title', 'company_profile', 'description', 'requirements',
for column in text_columns:
```

```
for column in text_columns:
    df[f'{column}_processed'] = df[column].apply(preprocess_text)

# Verify the cleaning worked
def check_cleaning(text_series):
    all_text = ' '.join(text_series)
    special_chars = re.findall(r'[^a-zA-Z\s]', all_text)
    remaining = Counter(special_chars).most_common()

    if remaining:
        print("Remaining special characters:")
        print(remaining)
    else:
        print("No special characters remaining!")

# Check the results
for column in text_columns:
    print(f"\nChecking {column}_processed:")
    check_cleaning(df[f'{column}_processed'])
```

```
Checking title_processed:
Remaining special characters:
[('_', 1)]

Checking company_profile_processed:
Remaining special characters:
[('_', 3827)]

Checking description_processed:
Remaining special characters:
[('_', 8945)]

Checking requirements_processed:
Remaining special characters:
[('_', 2901)]

Checking benefits_processed:
Remaining special characters:
[('_', 3683)]
```

In [39]:

```
# Check text lengths again
for column in text_columns:
    avg_length_before = df[column].str.len().mean()
    avg_length_after = df[f'{column}_processed'].str.len().mean()

    print(f"\n{column}:")
    print(f"Average length before: {avg_length_before:.2f}")
    print(f"Average length after: {avg_length_after:.2f}")
    print(f"Retention rate: {(avg_length_after/avg_length_before)*100:.2f
```

```
title:
Average length before: 28.53
Average length after: 26.42
Retention rate: 92.59%

company_profile:
Average length before: 620.90
Average length after: 443.07
Retention rate: 71.36%
```

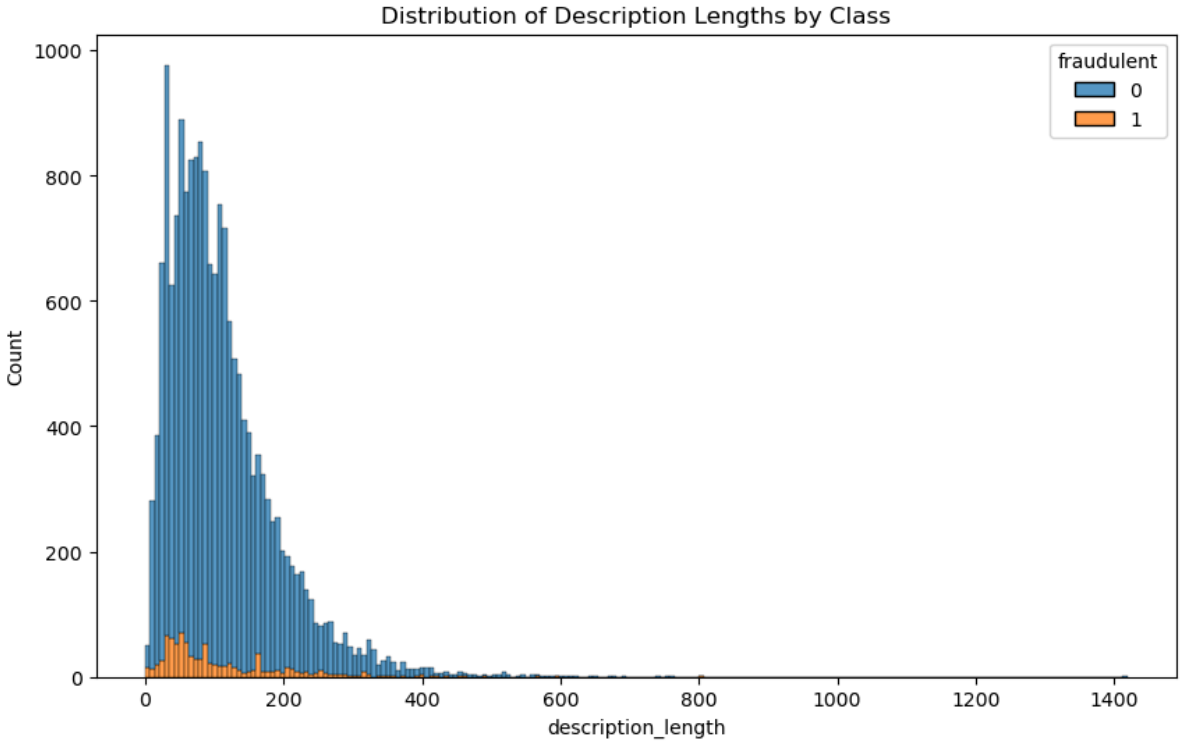
```
description:
Average length before: 1218.00
Average length after: 898.19
Retention rate: 73.74%

requirements:
Average length before: 590.13
Average length after: 464.45
Retention rate: 78.70%

benefits:
Average length before: 208.90
Average length after: 153.40
Retention rate: 73.43%
```

In [48]:

```
# Visualize the difference between the length of description for fraudule
plt.figure(figsize=(10, 6))
sns.histplot(df, x='description_length', hue='fraudulent', multiple='stacl
plt.title('Distribution of Description Lengths by Class')
plt.show()
```



Fradulent job postings have shorter descriptions than real job postings.

Text Vectorization

In [50]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [42]:

```
df.head()
```

Out[42]:

	title	department	company_profile	description	requir

0	Marketing Intern	Marketing	We're Food52, and we've created a groundbreaki...	Food52, a fast-growing, James Beard Award-winn...	Experie content mana syster
1	Customer Service - Cloud Video Production	Success	90 Seconds, the worlds Cloud Video Production ...	Organised - Focused - Vibrant - Awesome!Do you...	What we exp you:` respo
2	Commissioning Machinery Assistant (CMA)		Valor Services provides Workforce Solutions th...	Our client, located in Houston, is actively se...	Implem commissio commissi
3	Account Executive - Washington DC	Sales	Our passion for improving quality of life thro...	THE COMPANY: ESRI – Environmental Systems Rese...	EDUCATION: Ba or Master'
4	Bill Review Manager		SpotSource Solutions LLC is a Global Human Cap...	JOB TITLE: Itemization Review ManagerLOCATION:...	QUALIFICATI license in the

5 rows × 21 columns

In [53]:
```
# Combine relevant text columns for vectorization
df['combined_text']= (df['title_processed'] + ' ' +
                      df['company_profile_processed'] + ' ' +
                      df['description_processed'] + ' ' +
                      df['requirements_processed'] + ' ' +
                      df['benefits_processed'])

vectorizer = TfidfVectorizer(min_df = 0.01, max_df = 0.95)
X_tfidf = vectorizer.fit_transform(df['combined_text'])

vec_tfidf = pd.DataFrame(X_tfidf.toarray(), columns=vectorizer.get_feature
vec_tfidf.head()
```

Out[53]:

	ability	able	abroad	academic	accept	access	accessible	accommodat
0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0	
1	0.011488	0.013888	0.0	0.0	0.0	0.0	0.0	
2	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0	
3	0.020885	0.000000	0.0	0.0	0.0	0.0	0.0	
4	0.000000	0.030983	0.0	0.0	0.0	0.0	0.0	

5 rows × 2472 columns

Topic Modeling

In [62]:
```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation

# Document-term matrix using CountVectorizer
count_vectorizer = CountVectorizer(min_df=0.01, max_df=0.95)
doc_term_matrix = count_vectorizer.fit_transform(df['combined_text'])

# FLDA model
# n_components represents the number of topics you want to extract
n_topics = 10
lda_model = LatentDirichletAllocation(n_components=n_topics,
                                      random_state=42,
                                      learning_method='batch')
lda_output = lda_model.fit_transform(doc_term_matrix)

# Display the top words for each topic
def print_topics(model, feature_names, n_top_words):
    for topic_idx, topic in enumerate(model.components_):
        top_words = [feature_names[i] for i in topic.argsort()[:-n_top_wo
        print(f"Topic {topic_idx + 1}: {', '.join(top_words)}")

# Display top 10 words for each topic
print_topics(lda_model, count_vectorizer.get_feature_names_out(), 10)

# Add topic distributions to original dataframe
topic_names = [f"Topic {i+1}" for i in range(n_topics)]
doc_topics = pd.DataFrame(lda_output, columns=topic_names)

# Get dominant topic for each document
df['dominant_topic'] = doc_topics.idxmax(axis=1)
```

```
Topic 1: company, finance, financial, service, benefit, employment, positio
n, year, well, credit
Topic 2: marketing, medium, digital, social, content, brand, campaign, onli
ne, experience, new
Topic 3: business, management, client, project, team, experience, skill, wo
rk, service, company
Topic 4: experience, job, technical, system, engineering, year, website, am
p, data, manufacturing
Topic 5: experience, product, design, development, software, technology, se
rvice, application, team, user
Topic 6: student, job, teacher, abroad, get, loan, teaching, required, expe
rience, title
Topic 7: customer, service, work, process, business, document, communicatio
n, solution, required, mail
Topic 8: sale, customer, product, service, business, work, career, role, ca
ndidate, training
Topic 9: service, work, home, time, care, must, experience, position, custo
mer, hour
Topic 10: team, work, experience, people, new, working, company, looking, p
roduct, want
```

In [55]:
```
df.shape
```

Out[55]: (17880, 22)

Dimensionality reduction with SVD

In [73]:
```
from sklearn.decomposition import PCA

X_tfidf.shape
```

Out[73]: (17880, 2472)

In [83]:
```

```

Out[83]:
```
<Compressed Sparse Row sparse matrix of dtype 'float64'
        with 2444006 stored elements and shape (17880, 2472)>
```

In [86]:
```
from sklearn.decomposition import TruncatedSVD

# Apply Truncated SVD to sparse TF-IDF matrix
svd = TruncatedSVD(n_components=100, random_state=42)
tfidf_reduced = svd.fit_transform(X_tfidf)
print(tfidf_reduced.shape)
print("Explained Variance:", sum(svd.explained_variance_ratio_))
```

```
(17880, 100)
Explained Variance: 0.3984247159529867
```

SVD is not able to explain enough variance so we'll use other techniques

In [88]:
```
# Creating cleaned dataframe with processed features and numerical feature
cleaned_df = df[['description_length','dominant_topic','telecommuting','ha

cleaned_df = pd.concat([cleaned_df, vec_tfidf], axis=1)
cleaned_df.shape
```

Out[88]: (17880, 2478)

In [89]:
```
cleaned_df.head()
```

Out[89]:

	description_length	dominant_topic	telecommuting	has_company_logo	has_ques
0	84	Topic 2	0	1	
1	194	Topic 2	0	1	
2	30	Topic 5	0	1	
3	225	Topic 3	0	1	
4	131	Topic 3	0	1	

5 rows × 2478 columns

In [90]:
```
cleaned_df.to_csv('/Users/sabrinasayed/Documents/GitHub/Fake-Job-Posts/Da
```