In [1]:
```
import warnings
warnings.filterwarnings('ignore')

import pandas as pd

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

In [2]:
```
df.head()
```

Out[2]:

	job_id	job_role	company	experience	salary	location	rat
0	7.012301e+10	Branch Banking - Calling For Women Candidates	Hdfc Bank	1-6 Yrs	Not disclosed	Kolkata, Hyderabad/Secunderabad, Pune, Ahmedab...	
1	6.012391e+10	Product Owner Senior Manager	Accenture	11-15 Yrs	Not disclosed	Kolkata, Mumbai, Hyderabad/Secunderabad, Pune,...	
2	6.012391e+10	Employee Relations and Policies Associate Manager	Accenture	3-7 Yrs	Not disclosed	Kolkata, Mumbai, Hyderabad/Secunderabad, Pune,...	
3	6.012391e+10	Employee Relations and Policies Specialist	Accenture	3-7 Yrs	Not disclosed	Kolkata, Mumbai, Hyderabad/Secunderabad, Pune,...	
4	6.012301e+10	SAP BO Consultant	Mindtree	5-7 Yrs	Not disclosed	Hybrid - Kolkata, Hyderabad/Secunderabad, Pune...	

In [3]:
```
df.isnull().sum()
```

Out[3]:
```
job_id              480
job_role            480
company             481
experience         1749
salary              480
location           1706
rating            36199
reviews           36199
resposibilities     500
posted_on           480
job_link            480
company_link        480
dtype: int64
```

In [4]:
```
# check for duplicates
df.duplicated(subset='job_id').sum()
```

Out[4]:
```
6137
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 79147 entries, 0 to 79146
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   job_id           78667 non-null  float64
 1   job_role         78667 non-null  object 
 2   company          78666 non-null  object 
 3   experience       77398 non-null  object 
 4   salary           78667 non-null  object 
 5   location         77441 non-null  object 
 6   rating           42948 non-null  float64
 7   reviews          42948 non-null  object 
 8   resposibilities  78647 non-null  object 
 9   posted_on        78667 non-null  object 
 10  job_link         78667 non-null  object 
 11  company_link     78667 non-null  object 
dtypes: float64(2), object(10)
memory usage: 7.2+ MB
```

In [6]:
```
df.columns = df.columns.str.strip()
```

In [7]:
```
print(df.columns.tolist())
```

```
['job_id', 'job_role', 'company', 'experience', 'salary', 'location', 'rating', 'rev
iews', 'resposibilities', 'posted_on', 'job_link', 'company_link']
```

In [8]:
```
#remove columns
df = df.drop(columns=['posted_on', 'job_link', 'company_link'])
```

2. Data Cleaning

2.1) Dealing with Null Values Removed Rows:

Entries with missing values in job_id, resposibilities, and company were completely removed. Imputed Values:

Experience: Missing values filled with -> 5-10 Yrs Location: Missing values filled with -> Bangalore/Bengaluru Rating: Missing values filled with -> 0.0 (indicating no rating) Reviews: Missing values filled with -> 0 Reviews These steps ensure a cleaner and more consistent dataset for analysis.

In [11]:
```
df.isnull().sum()

df = df.dropna(subset =['job_id','resposibilities','company'])

df['experience'] = df['experience'].fillna('5-10 Yrs')
df['location'] = df['location'].fillna('Bangalore/Bengaluru')
df['rating'] = df['rating'].fillna(0.0)
df['reviews'] = df['reviews'].fillna('0 Reviews')
```

In [12]:
```
df = df.drop_duplicates(subset = 'job_id')
```

In [ ]:
```
# change the data type of job_id to string
df['job_id'] = df['job_id'].astype(str)
```

In [14]:
```
df['min_experience'] = df['experience'].str.replace('Yrs','').str.split('-').str[0]
df['max_experience'] = df['experience'].str.replace('Yrs','').str.split('-').str[1]
```

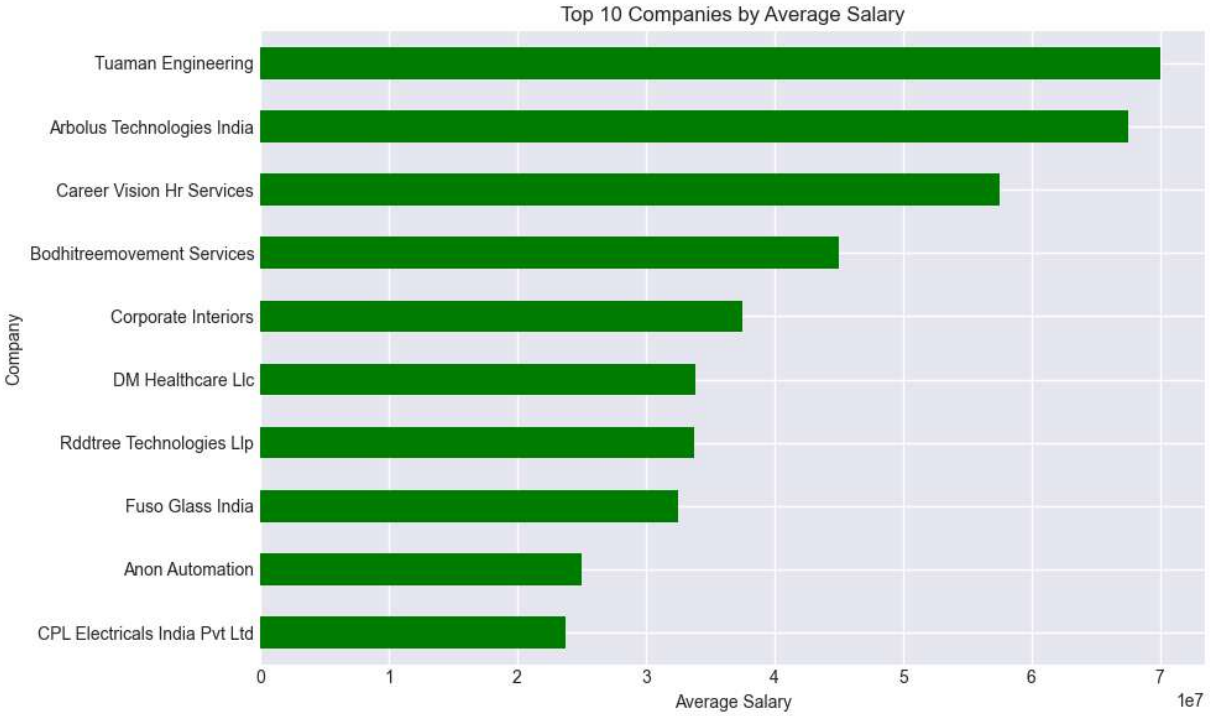
Clean Salary Column

In [15]:
```
df['clean_salary'] = df['salary'].str.replace(' PA.','').str.replace(',','').str.sp

df = df[(df['clean_salary'] != '9.5 Cr and above') & (df['clean_salary'] != 'Less t
df['min_salary'] = df['clean_salary'].str.split('(').str[0].str.strip()
del df['clean_salary']


df['clean_salary'] = df['salary'].str.replace(' PA.','').str.replace(',','').str.sp
df = df[df['clean_salary'] != '9.5 Cr and above']
df['max_salary'] = df['clean_salary'].fillna('Not disclosed')
del df['clean_salary']

del df['salary']
del df['experience']
```

Salary Distribution by Company Find which companies offer the highest average salaries.

In [23]:
```
df['min_salary'] = pd.to_numeric(df['min_salary'], errors='coerce')
df['max_salary'] = pd.to_numeric(df['max_salary'], errors='coerce')
df['avg_salary'] = (df['min_salary'] + df['max_salary']) / 2

top_salary_companies = (
    df[df['avg_salary'].notna()]
    .groupby('company')['avg_salary']
```

```
    .mean()
    .sort_values(ascending=False)
    .head(10)
)

top_salary_companies.plot(kind='barh', figsize=(10, 6), color='green')
plt.title("Top 10 Companies by Average Salary")
plt.xlabel("Average Salary")
plt.ylabel("Company")
plt.gca().invert_yaxis()
plt.grid(True)
plt.tight_layout()
plt.show()
```



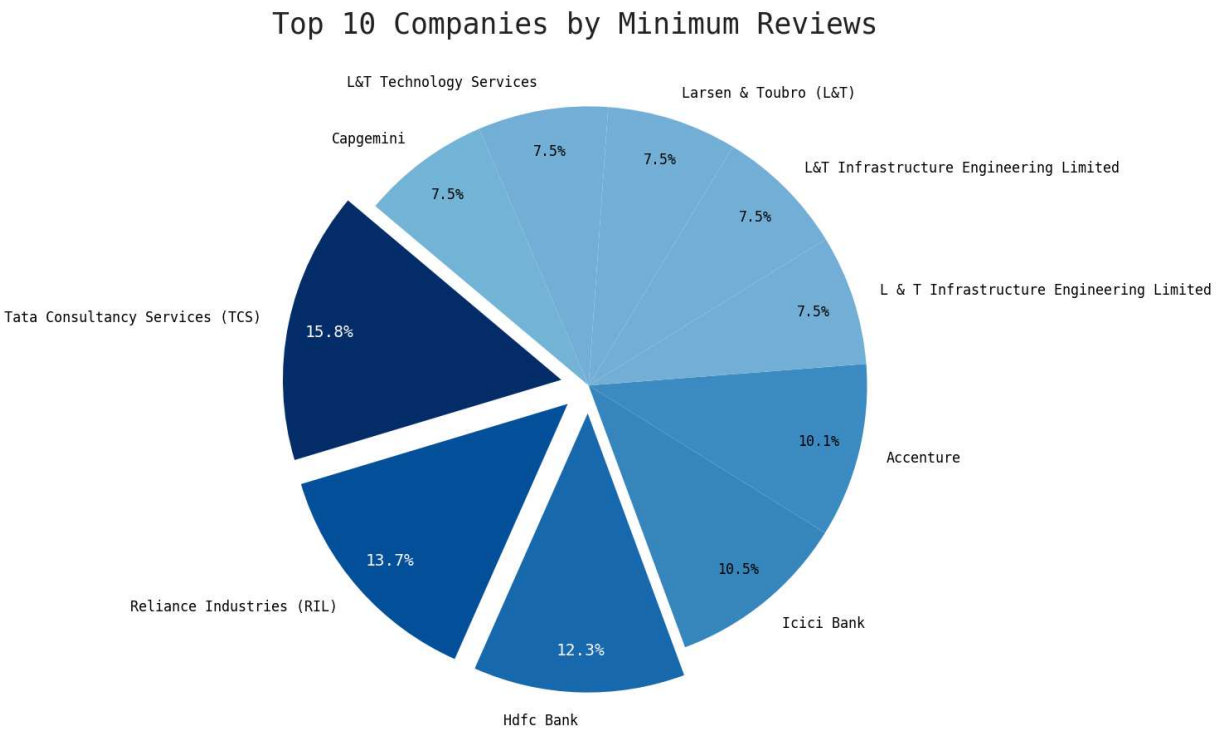
In [16]:
```
df['reviews'] = df['reviews'].str.split(' ').str[0].astype('int')
```

In [17]:
```
print('-'*30)
print('Total Jobs         : ', len(df))
print('Total Companies    : ', df['company'].nunique())
print('-'*30)
```

```
------------------------------
Total Jobs         :  72967
Total Companies    :  15310
------------------------------
```

In [18]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import numpy as np

# Assuming you want to plot the same data
company_reviews = df.groupby('company')['reviews'].min().sort_values(ascending=Fals
```

```
# Seaborn typically works well with dataframes, so let's create one from the series
company_reviews_df = company_reviews.reset_index()
company_reviews_df.columns = ['company', 'reviews']

# Define a colormap (e.g., Blues)
colors = cm.Blues(company_reviews_df['reviews'] / float(max(company_reviews_df['rev

plt.figure(figsize=(10, 10))

# Seaborn doesn't have a direct pie chart function like matplotlib.
# We will use matplotlib's pie function but incorporate seaborn styling.
plt.style.use('seaborn-v0_8-darkgrid') # Apply a seaborn style

# Create an explode list: Explode the first 3, rest are 0
explode = [0.1 if i < 3 else 0 for i in range(len(company_reviews_df))]

wedges, texts, autotexts = plt.pie(
    company_reviews_df['reviews'],
    labels=company_reviews_df['company'],
    autopct='%1.1f%%',
    startangle=140,
    colors=colors,
    textprops=dict(color="black", fontsize=12, fontname='monospace'),
    pctdistance=0.85,
    explode=explode
)

# Identify the indices of the top 3 values
top3_indices = company_reviews_df['reviews'].nlargest(3).index

# Set the color and font for the autotexts (percentage annotations) for the top 3 s
for i, review_count in enumerate(company_reviews_df['reviews']):
    if i in top3_indices:
        autotexts[i].set_color('white')
        autotexts[i].set_fontsize(14)

# Set font size for all other annotations
for i, review_count in enumerate(company_reviews_df['reviews']):
    if i not in top3_indices:
        autotexts[i].set_fontsize(12)


plt.title('Top 10 Companies by Minimum Reviews', fontsize=25, fontname='monospace',
plt.axis('equal')
plt.show()
```



Top 10 Companies by Minimum Reviews Analysis Overview This analysis focuses on identifying the top 10 companies with the highest minimum number of reviews from job postings. The goal is to highlight major industry players that have garnered significant user feedback, indicating their strong presence in the job market.

Key Insights Tata Consultancy Services (TCS) dominates the list with 15.8% of the total review share, showcasing its large-scale hiring and market presence. Reliance Industries (RIL) holds the second position with 13.7%, reflecting its expansive recruitment strategy. HDFC Bank secures the third spot with 12.3%, indicating its robust hiring processes. ICICI Bank and Accenture follow with 10.5% and 10.1% respectively, highlighting their competitive edge in the market. Companies like Capgemini, L&T Technology Services, and Larsen & Toubro (L&T) also feature prominently, each holding 7.5% of the review distribution. Business Implications Market Leaders: The analysis underscores TCS, Reliance, and HDFC Bank as market leaders in terms of visibility and review engagement. Recruitment Strength: High review volumes reflect large-scale recruitment and strong market presence. Competitive Landscape: The distribution provides insights into competitive hiring strategies across major industries. Future Scope Region-wise Analysis: Further analysis can be done to map these companies' recruitment presence across different cities. Role-based Segmentation: Breaking down the reviews by job roles to identify the most common positions offered. Skill Mapping: Exploring the specific skills these companies are targeting in their recruitment. Conclusion This visualization offers a clear snapshot of the Top 10 companies leading the market based on review counts, reflecting their strong brand visibility and recruitment effectiveness. These insights are crucial for job seekers, market analysts, and recruiters to understand competitive hiring trends.

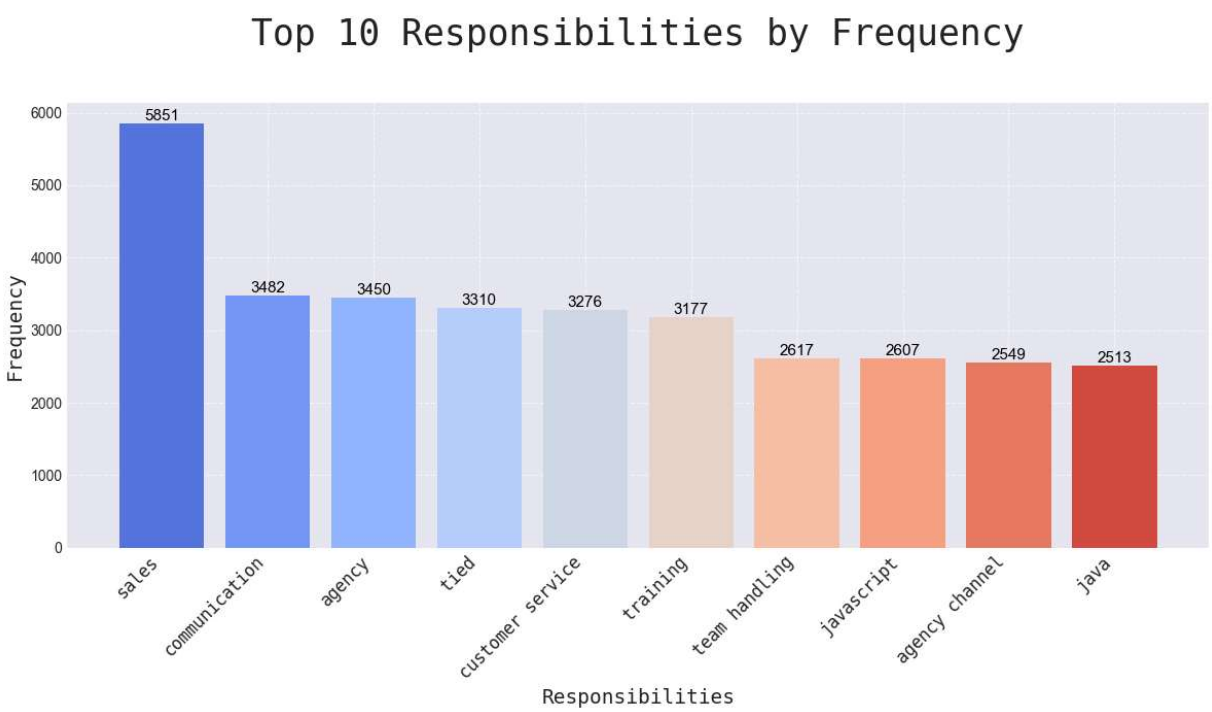
3.3 Companies Hiring for Data Analyst

In [19]:
```
len(df[df['job_role'] == 'Data Analyst'])
```

Out[19]: 59

3.4 Skills needed for almost all the jobs

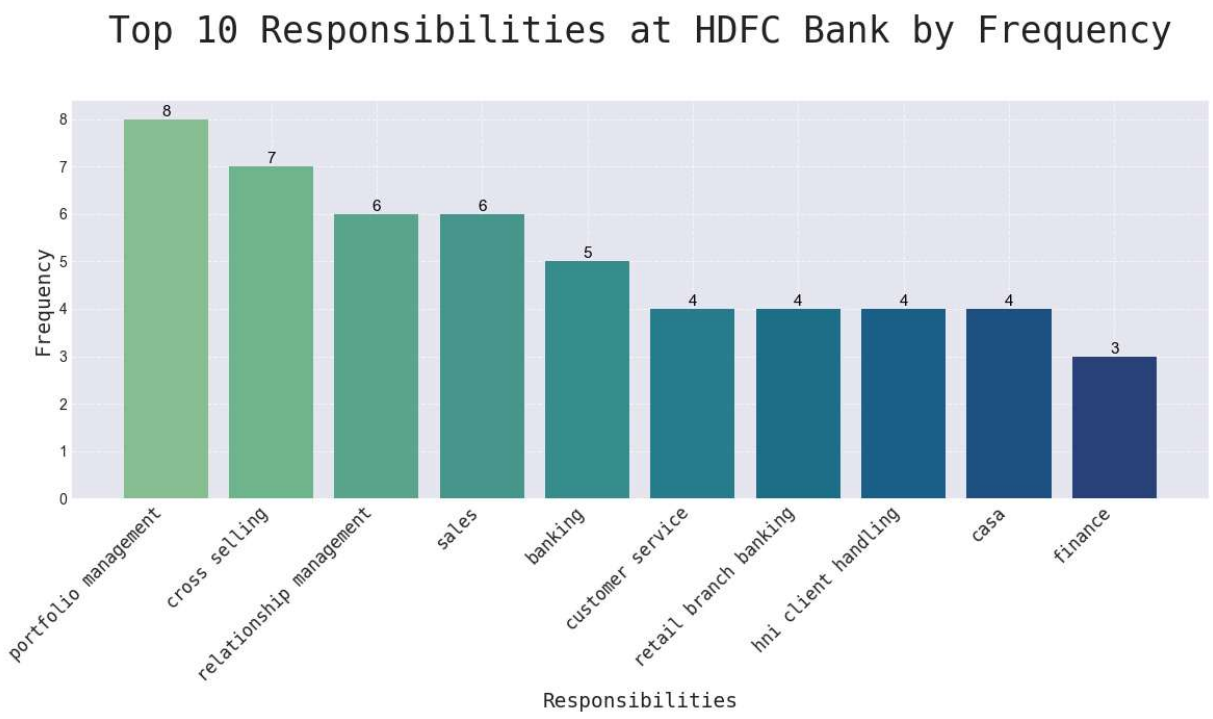
In [20]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming 'df' is your DataFrame
responsibility_counts = df['resposibilities'].str.lower().str.split(',').explode().

# Set Seaborn style
plt.style.use('seaborn-v0_8-darkgrid')

# Create the bar chart
plt.figure(figsize=(12, 7))
bars = plt.bar(responsibility_counts.index, responsibility_counts.values,
               color=sns.color_palette("coolwarm", len(responsibility_counts)))

# Add title and labels
plt.title('Top 10 Responsibilities by Frequency', fontsize=25, fontname='monospace'
plt.xlabel('Responsibilities', fontsize=14, fontname='monospace')
plt.ylabel('Frequency', fontsize=14, fontname='monospace')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45, ha='right', fontsize=12, fontname='monospace')

# Annotate each bar with its value
for bar, value in zip(bars, responsibility_counts.values):
    plt.text(bar.get_x() + bar.get_width() / 2, value, f'{value}',
             ha='center', va='bottom', fontsize=11, color='black')

# Add gridlines for better readability
plt.grid(True, linestyle='--', alpha=0.7)

# Display the plot
plt.tight_layout()
plt.show()
```



3.5 Skills to get hired in HDFC Bank

In [21]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Filter data for HDFC Bank and get top 10 responsibilities
hdfc_responsibilities = (
    df[df['company'] == 'Hdfc Bank']['resposibilities']
    .str.lower()
    .str.split(',')
    .explode()
    .value_counts()
    .head(10)
)

# Set Seaborn style
plt.style.use('seaborn-v0_8-darkgrid')

# Create the bar chart
plt.figure(figsize=(12, 7))
bars = plt.bar(hdfc_responsibilities.index, hdfc_responsibilities.values,
               color=sns.color_palette("crest", len(hdfc_responsibilities)))

# Add title and labels
plt.title('Top 10 Responsibilities at HDFC Bank by Frequency', fontsize=25, fontnam
plt.xlabel('Responsibilities', fontsize=14, fontname='monospace')
plt.ylabel('Frequency', fontsize=14, fontname='monospace')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45, ha='right', fontsize=12, fontname='monospace')

# Annotate each bar with its value
for bar, value in zip(bars, hdfc_responsibilities.values):
    plt.text(bar.get_x() + bar.get_width() / 2, value, f'{value}',
```

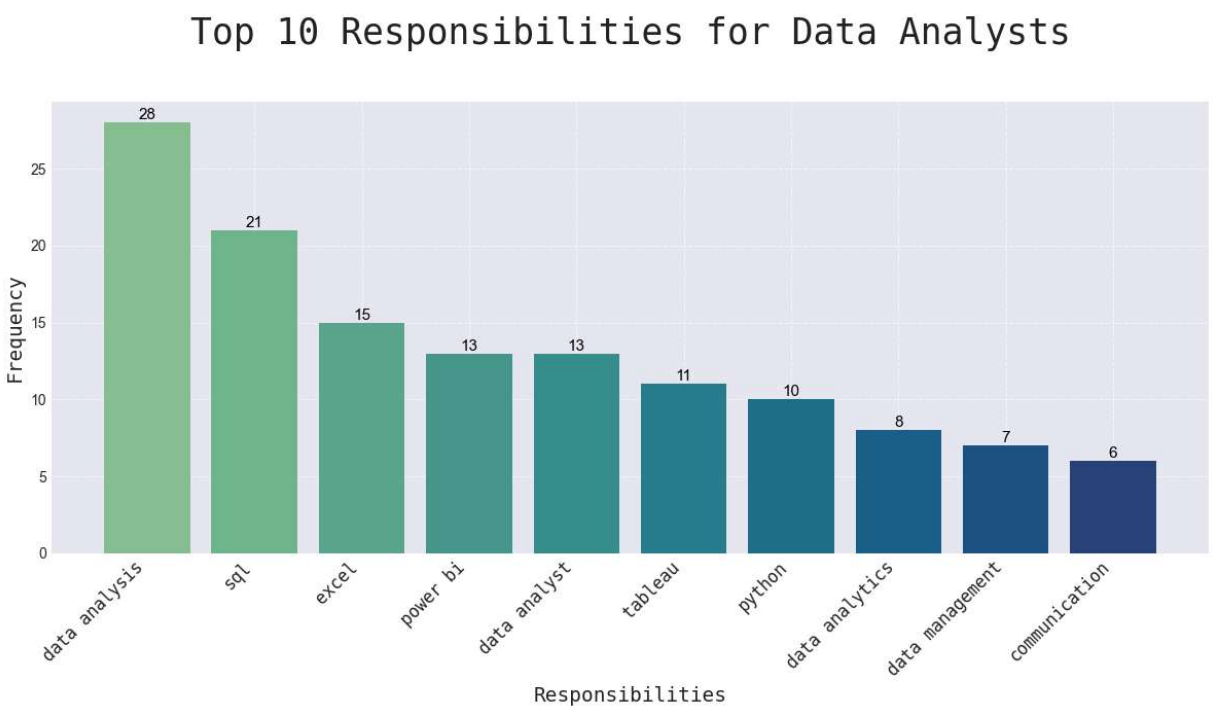
```
            ha='center', va='bottom', fontsize=11, color='black')

# Add gridlines for better readability
plt.grid(True, linestyle='--', alpha=0.7)

# Display the plot
plt.tight_layout()
plt.show()
```



3.6 Skills Needed to become a Data Analyst

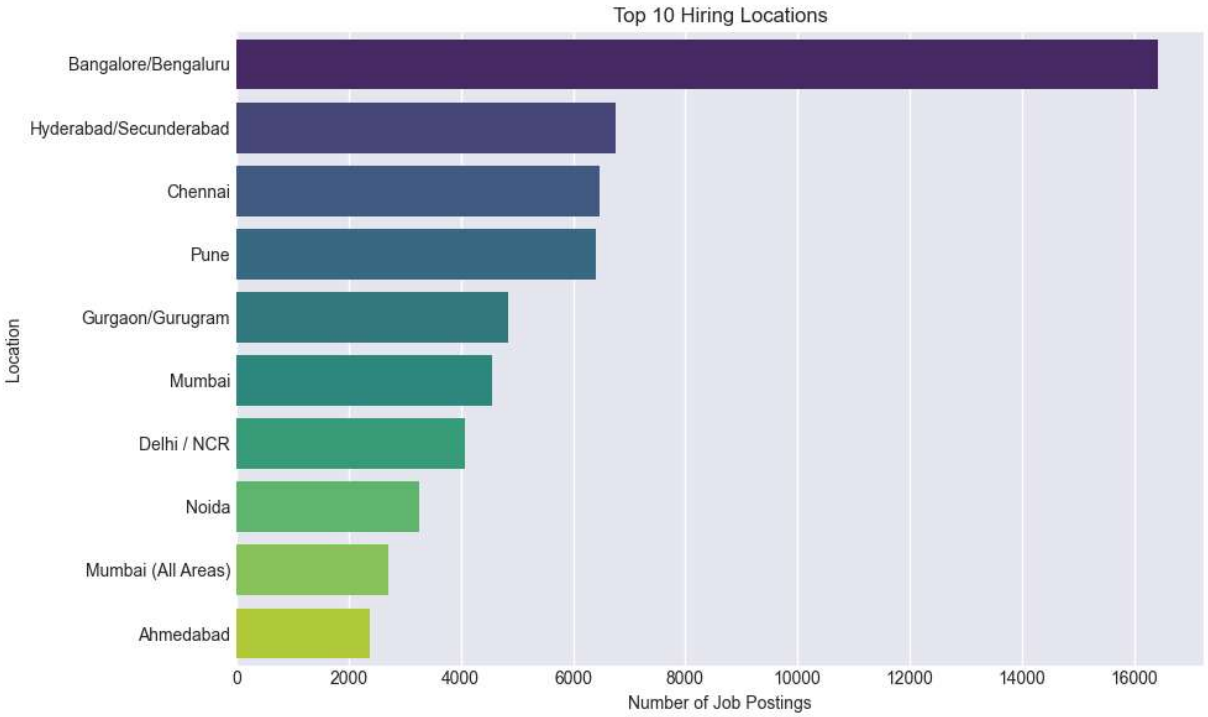
In [22]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Filter data for Data Analyst role and get top 10 responsibilities
data_analyst_responsibilities = (
    df[df['job_role'] == 'Data Analyst']['resposibilities']
    .str.lower()
    .str.split(',')
    .explode()
    .value_counts()
    .head(10)
)

# Set Seaborn style
plt.style.use('seaborn-v0_8-darkgrid')

# Create the bar chart
plt.figure(figsize=(12, 7))
bars = plt.bar(data_analyst_responsibilities.index, data_analyst_responsibilities.v
               color=sns.color_palette("crest", len(data_analyst_responsibilities))

# Add title and labels
plt.title('Top 10 Responsibilities for Data Analysts', fontsize=25, fontname='monos
```

```
plt.xlabel('Responsibilities', fontsize=14, fontname='monospace')
plt.ylabel('Frequency', fontsize=14, fontname='monospace')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45, ha='right', fontsize=12, fontname='monospace')

# Annotate each bar with its value
for bar, value in zip(bars, data_analyst_responsibilities.values):
    plt.text(bar.get_x() + bar.get_width() / 2, value, f'{value}',
             ha='center', va='bottom', fontsize=11, color='black')

# Add gridlines for better readability
plt.grid(True, linestyle='--', alpha=0.7)

# Display the plot
plt.tight_layout()
plt.show()
```



Top Hiring Locations

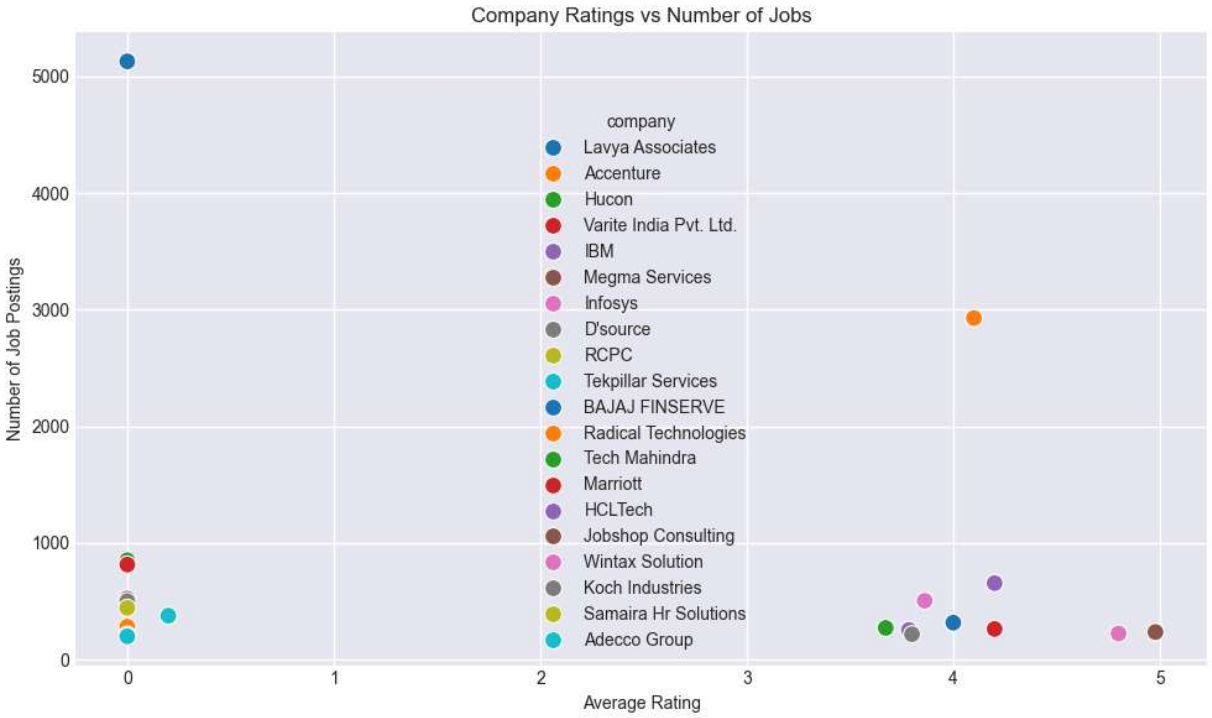
In [24]:
```
top_locations = df['location'].str.split(',').explode().str.strip().value_counts().

plt.figure(figsize=(10,6))
sns.barplot(x=top_locations.values, y=top_locations.index, palette='viridis')
plt.title('Top 10 Hiring Locations')
plt.xlabel('Number of Job Postings')
plt.ylabel('Location')
plt.tight_layout()
plt.show()
```



Company-Level Trends

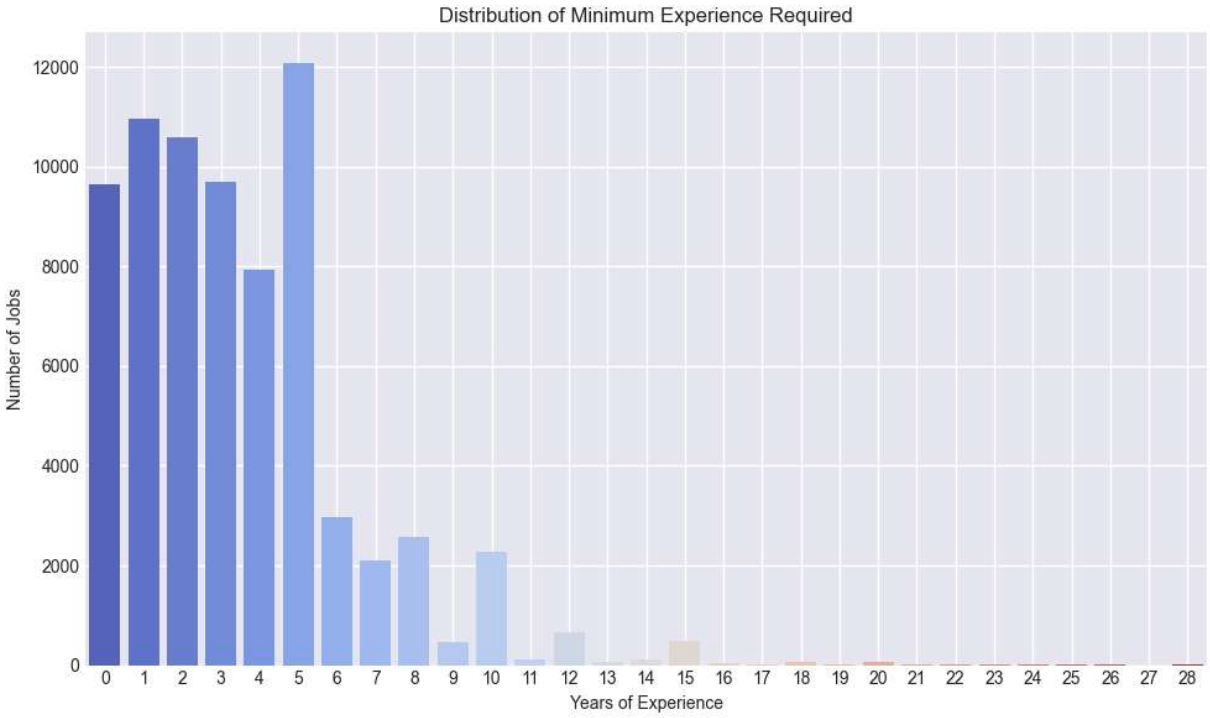
Hiring Frequency vs Rating

In [47]:
```
company_metrics = df.groupby('company').agg({
    'job_id': 'count',
    'rating': 'mean'
}).rename(columns={'job_id': 'num_jobs'}).sort_values('num_jobs', ascending=False).
plt.figure(figsize=(10,6))
sns.scatterplot(x='rating', y='num_jobs', data=company_metrics, hue=company_metrics
plt.title("Company Ratings vs Number of Jobs")
plt.xlabel("Average Rating")
plt.ylabel("Number of Job Postings")
plt.grid(True)
plt.tight_layout()
plt.show()
```



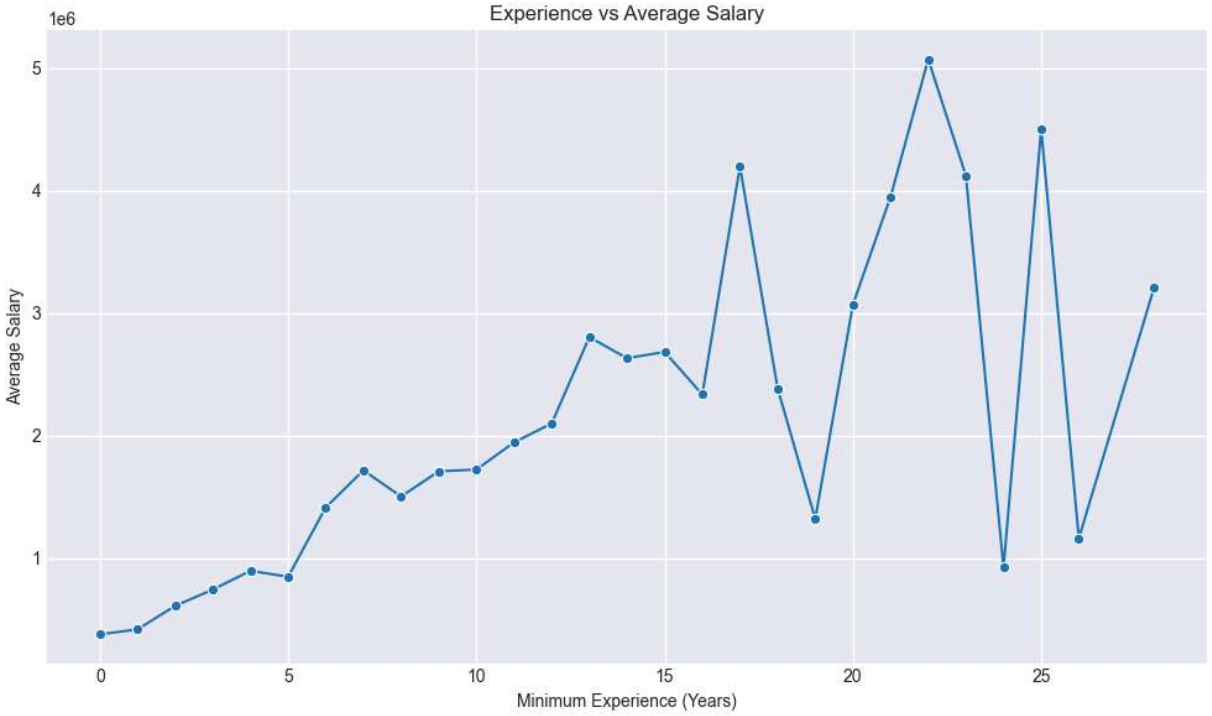
Experience-Level Analysis Distribution of Experience Required

In [26]:
```
exp_distribution = df['min_experience'].value_counts().sort_index()

plt.figure(figsize=(10,6))
sns.barplot(x=exp_distribution.index, y=exp_distribution.values, palette='coolwarm'
plt.title('Distribution of Minimum Experience Required')
plt.xlabel('Years of Experience')
plt.ylabel('Number of Jobs')
plt.grid(True)
plt.tight_layout()
plt.show()
```



Experience vs Salary Trend

In [27]:
```
exp_salary = df.groupby('min_experience')['avg_salary'].mean()

plt.figure(figsize=(10,6))
sns.lineplot(x=exp_salary.index, y=exp_salary.values, marker='o')
plt.title("Experience vs Average Salary")
plt.xlabel("Minimum Experience (Years)")
plt.ylabel("Average Salary")
plt.grid(True)
plt.tight_layout()
plt.show()
```



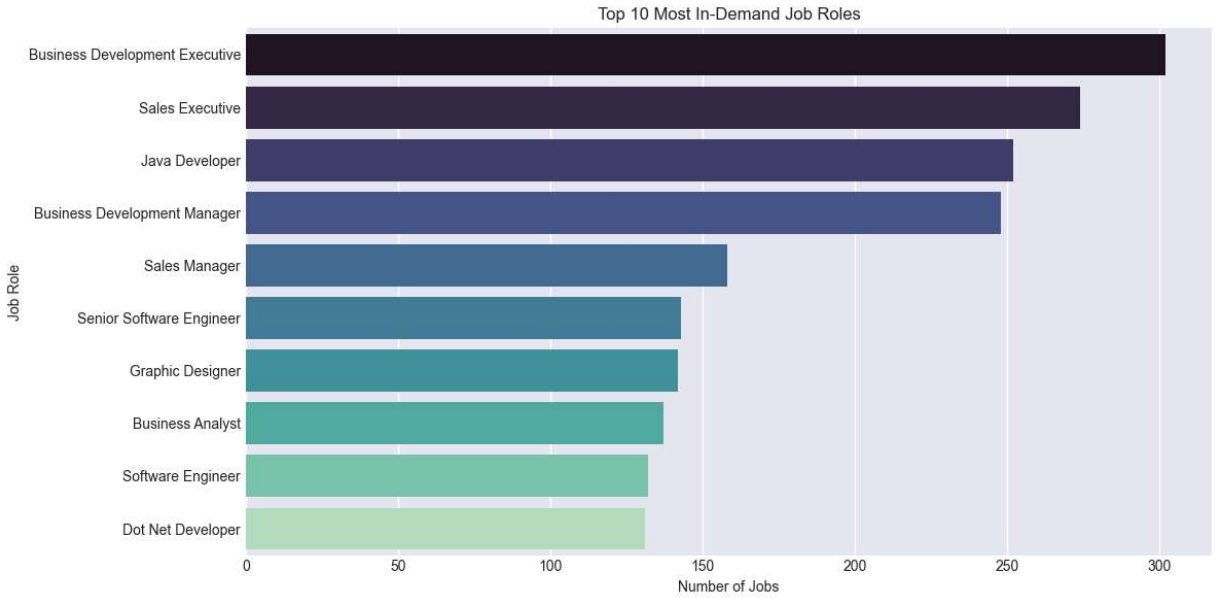
Keyword or Skill Trends (Text Analysis)

WordCloud for Responsibilities

In [37]:
```
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Combine all responsibilities into one string
text = ' '.join(df['resposibilities'].str.lower().values)

# Generate the word cloud
wordcloud = WordCloud(
    width=1000,
    height=500,
    background_color='white',
    colormap='plasma'
).generate(text)

# Display the word cloud
plt.figure(figsize=(15, 7))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Most Common Skills/Keywords in Job Responsibilities', fontsize=20, fontn
plt.tight_layout()
plt.show()
```

Most In-Demand Job Roles

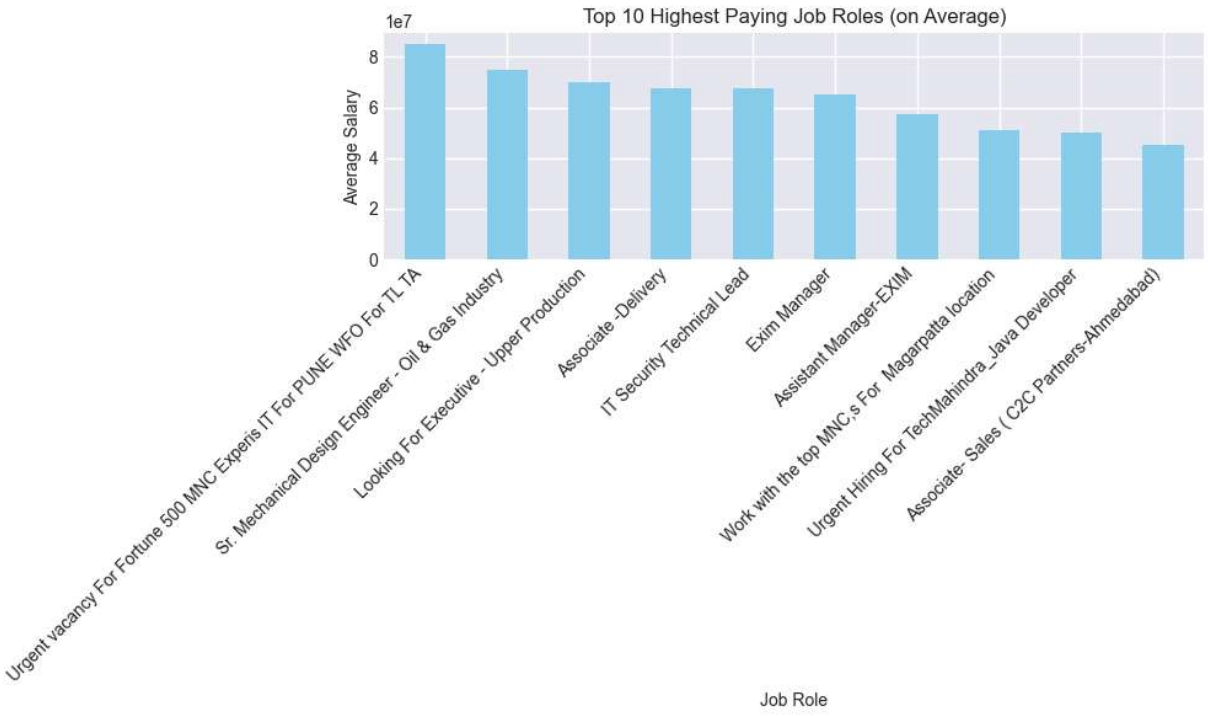
In [31]:
```
job_counts = df['job_role'].value_counts().head(10)

plt.figure(figsize=(12,6))
sns.barplot(x=job_counts.values, y=job_counts.index, palette='mako')
plt.title('Top 10 Most In-Demand Job Roles')
plt.xlabel('Number of Jobs')
plt.ylabel('Job Role')
plt.tight_layout()
plt.show()
```



Average Salary by Job Role

In [32]:
```
role_salary = (
    df.groupby('job_role')['avg_salary']
```

```
    .mean()
    .sort_values(ascending=False)
    .head(10)
)

role_salary.plot(kind='bar', figsize=(10, 6), color='skyblue')
plt.title("Top 10 Highest Paying Job Roles (on Average)")
plt.ylabel("Average Salary")
plt.xlabel("Job Role")
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



Time Series (if posted_on was kept and cleaned) You mentioned posted_on was dropped — if you keep it and parse to date, you can analyze hiring trends over time:

In [41]:
```
# Strip all column names of whitespace
df.columns = df.columns.str.strip()
```