Project Name - Data Analyst Jobs (ML _ FA _ DA projects) (Part 1)

Project Type - Data Analysis

Industry - Unified Mentor

Contribution - Individual

Member Name - Hare Krishana Mishra

Task - 1

Project Summary -

Project Description:

This project analyzes over 2,000 job listings for Data Analyst roles to uncover industry trends, key skills, and salary patterns. Using data cleaning, exploratory data analysis (EDA), and machine learning, the project identifies factors influencing salary, such as company rating, size, sector, and required skills. A Random Forest Regressor model is trained to predict the average salary for given job attributes, helping job seekers and recruiters make data-driven career decisions.

Objective:

- To explore and visualize trends in Data Analyst job postings.
- To identify skills and company characteristics that significantly impact salaries.
- To develop a machine learning model that predicts average salaries based on job-related factors

Key Project Details:

Dataset: 2,253 job postings with details such as Salary Estimate, Company Rating, Location, Industry, and Job Description.

Data Cleaning: Removed duplicates, handled missing values, extracted numerical salary ranges, and standardized categorical values.

Feature Engineering:

Extracted technical skills (Python, Excel, SQL) from job descriptions.

Encoded categorical variables for machine learning compatibility.

EDA Insights:

Top-paying sectors include Biotech & Pharmaceuticals, Real Estate, and Arts & Entertainment.

California locations dominate the highest average salaries.

Larger companies don't always pay more than smaller companies.

Model:

Random Forest Regressor trained to predict salaries.

Evaluation metrics: Mean Absolute Error (MAE) and R^2 Score.

Feature importance analysis to understand key drivers of salary.

Outcome:

A predictive model and visual insights that can guide job seekers, HR professionals, and analysts in understanding market trends.

Let's Begin:-

Data Collection

In [32]:

```
import pandas as pd
```

In [33]:

```
# Load the dataset
data = pd.read_csv("/content/DataAnalyst.csv")
```

In [34]:

```
# Inspect the dataset
data.head()
```

Out[34]:

	Unnamed: 0	Job Title	Salary Estimate	Job Description	Rating	Company Name	Location	He
0	0	Data Analyst, Center on Immigration and Justic...	$37K-$ 66K (Glassdoor est.)	Are you eager to roll up your sleeves and harn...	3.2	Vera Institute of Justice\n3.2	New York, NY	Ne
1	1	Quality Data Analyst	$37K-$ 66K (Glassdoor est.)	Overview\n\nProvides analytical and technical ...	3.8	Visiting Nurse Service of New York\n3.8	New York, NY	Ne
2	2	Senior Data Analyst, Insights & Analytics Team...	$37K-$ 66K (Glassdoor est.)	We're looking for a Senior Data Analyst who ha...	3.4	Squarespace\n3.4	New York, NY	Ne
3	3	Data Analyst	$37K-$ 66K (Glassdoor est.)	Requisition NumberRR-0001939\nRemote:Yes\nWe c...	4.1	Celerity\n4.1	New York, NY	N

	Unnamed: 0	Job Title	Salary Estimate	Job Description	Rating	Company Name	Location	He
4	4	Reporting Data Analyst	$37K-$ 66K (Glassdoor est.)	ABOUT FANDUEL GROUP\n\nFanDuel Group is a worl...	3.9	FanDuel\n3.9	New York, NY	Ne

In [35]:

```
print(data.info())
```

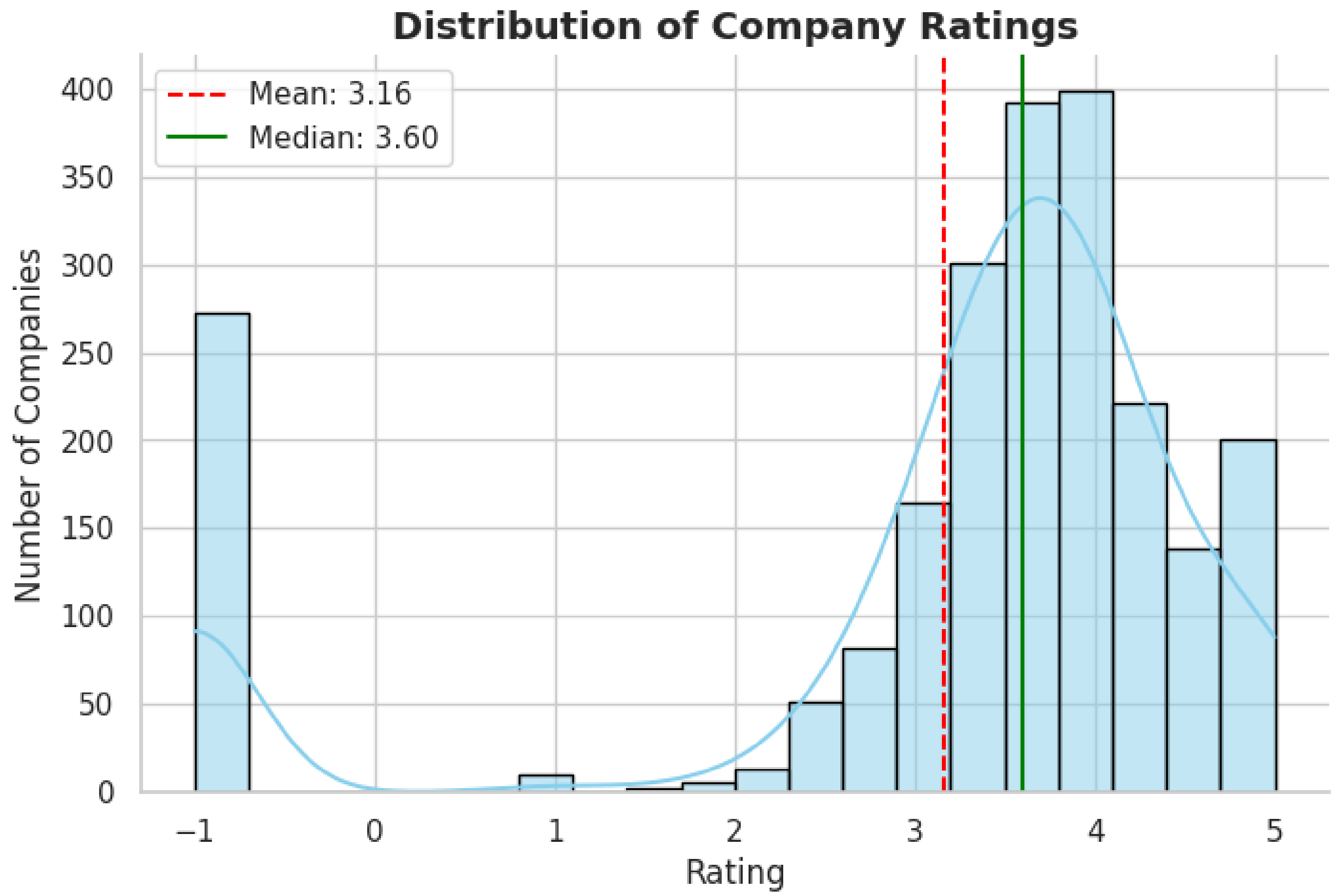
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2253 entries, 0 to 2252
Data columns (total 16 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Unnamed: 0         2253 non-null   int64  
 1   Job Title          2253 non-null   object 
 2   Salary Estimate    2253 non-null   object 
 3   Job Description    2253 non-null   object 
 4   Rating             2253 non-null   float64
 5   Company Name       2252 non-null   object 
 6   Location           2253 non-null   object 
 7   Headquarters       2253 non-null   object 
 8   Size               2253 non-null   object 
 9   Founded            2253 non-null   int64  
 10  Type of ownership  2253 non-null   object 
 11  Industry           2253 non-null   object 
 12  Sector             2253 non-null   object 
 13  Revenue            2253 non-null   object 
 14  Competitors        2253 non-null   object 
 15  Easy Apply         2253 non-null   object 
dtypes: float64(1), int64(2), object(13)
memory usage: 281.8+ KB
None
```

Distribution of Company Ratings

In [36]:

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(8, 5))
sns.histplot(data=data, x='Rating', bins=20, kde=True, color='skyblue', edgecolor='black

# Add mean and median lines
mean_rating = data['Rating'].mean()
median_rating = data['Rating'].median()
plt.axvline(mean_rating, color='red', linestyle='--', linewidth=1.5, label=f"Mean: {mean
plt.axvline(median_rating, color='green', linestyle='-', linewidth=1.5, label=f"Median:

# Styling
plt.title('Distribution of Company Ratings', fontsize=14, fontweight='bold')
plt.xlabel('Rating', fontsize=12)
plt.ylabel('Number of Companies', fontsize=12)
plt.legend()
sns.despine(top=True, right=True)
```

```
plt.show()
```



Exploratory Data Analysis (EDA)

In [37]:

```
# Check for duplicates
print(f"Duplicate rows: {data.duplicated().sum()}")
```

```
Duplicate rows: 0
```

In [38]:

```
# General statistics
data.describe(include='all')
```

Out[38]:

	Unnamed: 0	Job Title	Salary Estimate	Job Description	Rating	Company Name	Location	Headquarters
count	2253.0000	2253	2253	2253	2253.000000	2252	2253	2253
unique	NaN	1272	90	2253	NaN	1513	253	483
top	NaN	Data Analyst	$41K-$ 78K (Glassdoor est.)	You.\n\nYou bring your body, mind, heart and s...	NaN	Staffigo Technical Services, LLC\n5.0	New York, NY	New York, NY
freq	NaN	405	57	1	NaN	58	310	206
mean	1126.0000	NaN	NaN	NaN	3.160630	NaN	NaN	NaN

	Unnamed: 0	Job Title	Salary Estimate	Job Description	Rating	Company Name	Location	Headquarters
std	650.5294	NaN	NaN	NaN	1.665228	NaN	NaN	NaN
min	0.0000	NaN	NaN	NaN	-1.000000	NaN	NaN	NaN
25%	563.0000	NaN	NaN	NaN	3.100000	NaN	NaN	NaN
50%	1126.0000	NaN	NaN	NaN	3.600000	NaN	NaN	NaN
75%	1689.0000	NaN	NaN	NaN	4.000000	NaN	NaN	NaN
max	2252.0000	NaN	NaN	NaN	5.000000	NaN	NaN	NaN

Visualization

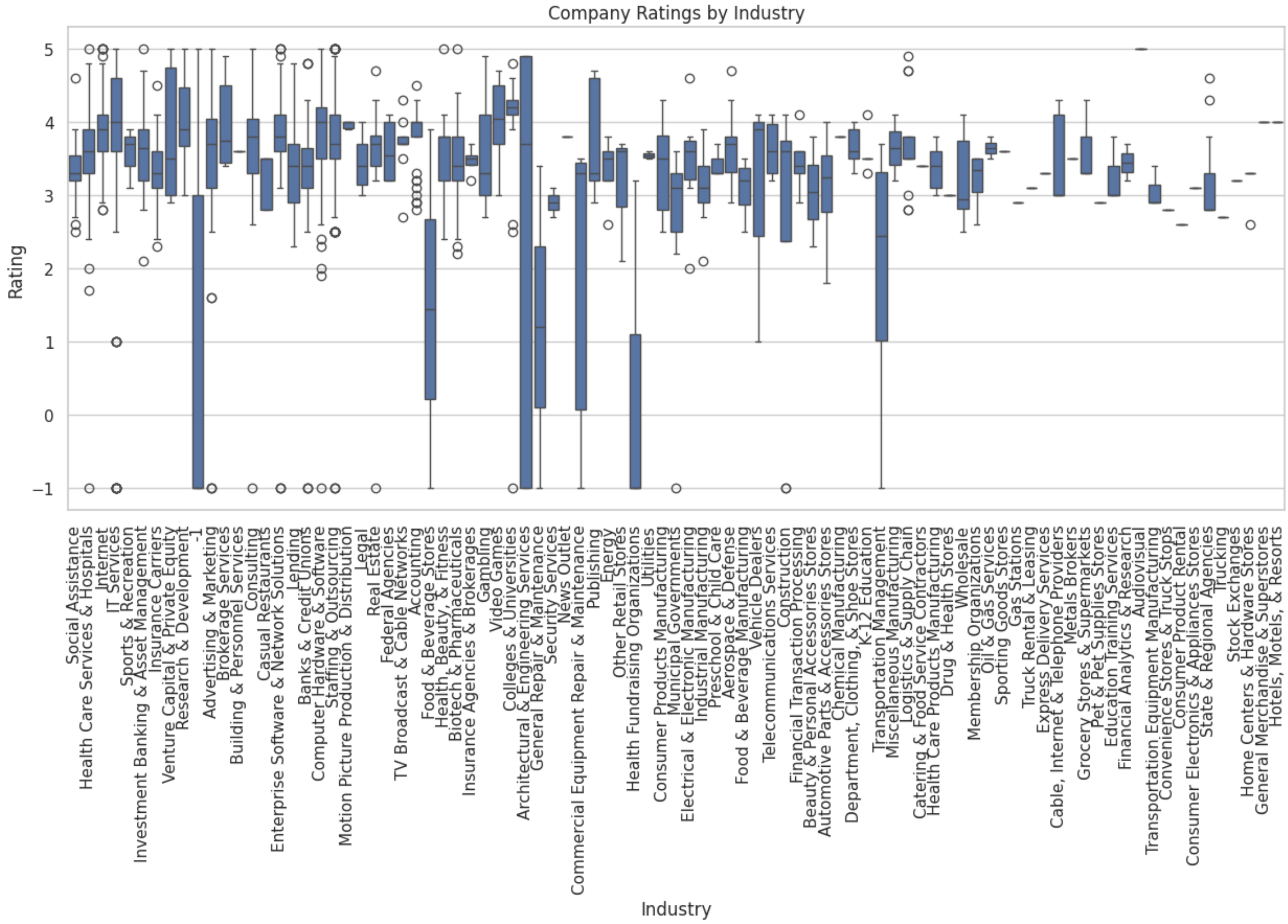
In [39]:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

Ratings by Industry

In [40]:

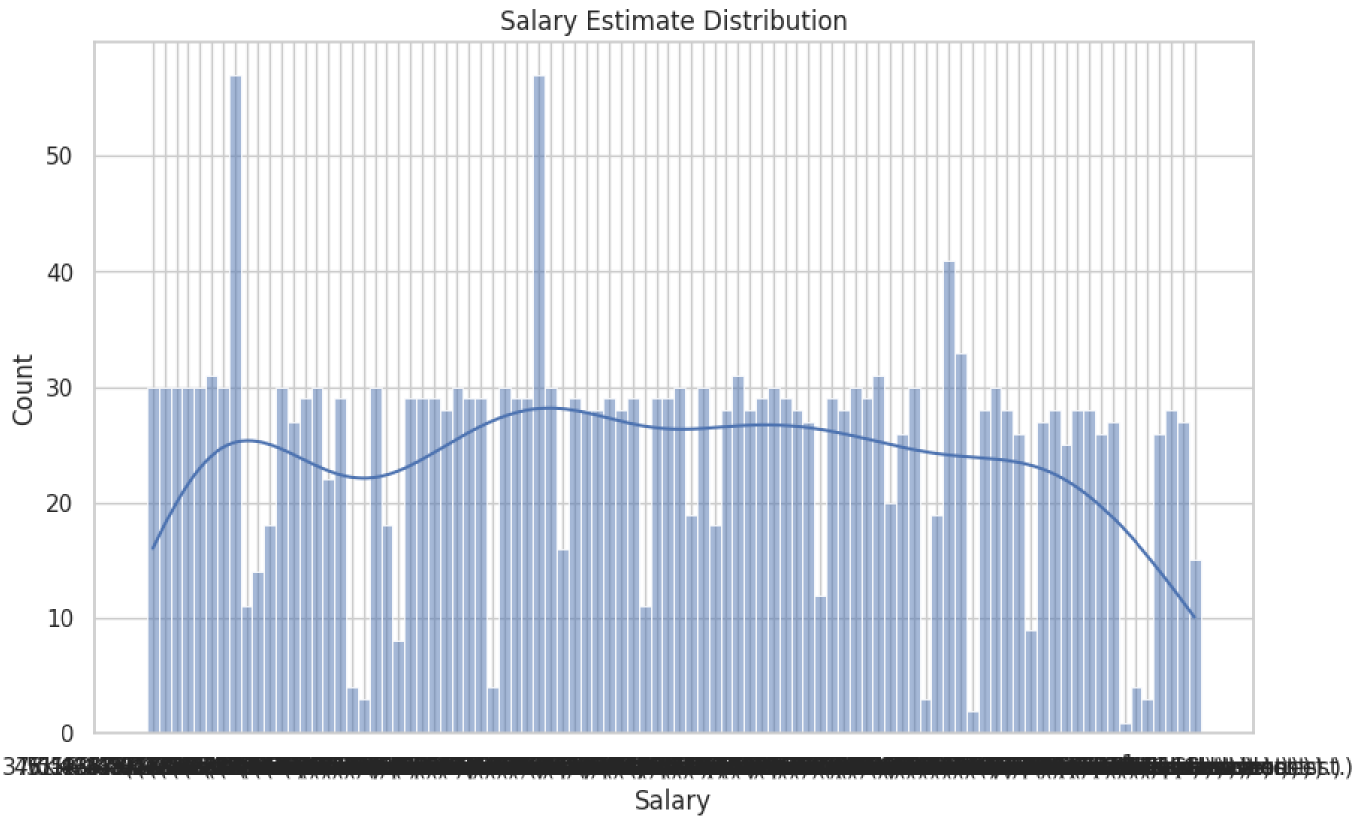
```
plt.figure(figsize=(15, 6))
sns.boxplot(x='Industry', y='Rating', data=data)
plt.xticks(rotation=90)
plt.title("Company Ratings by Industry")
plt.show()
```



Salary Distribution

In [41]:

```
# Salary distribution
plt.figure(figsize=(10, 6))
sns.histplot(data['Salary Estimate'], kde=True, bins=20)
plt.title("Salary Estimate Distribution")
plt.xlabel("Salary")
plt.show()
```



Data Cleaning

In [42]:

```
# Check missing values
print(data.isnull().sum())
```

```
Unnamed: 0           0
Job Title            0
Salary Estimate      0
Job Description      0
Rating               0
Company Name         1
Location             0
Headquarters         0
Size                 0
Founded              0
Type of ownership    0
Industry             0
Sector               0
Revenue              0
Competitors          0
Easy Apply           0
dtype: int64
```

In [43]:

```
# Fill missing numerical values
data['Rating'].fillna(data['Rating'].median(), inplace=True)
```

```
/tmp/ipython-input-1749438820.py:2: FutureWarning:

A value is trying to be set on a copy of a DataFrame or Series through chained assignmen
t using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the
intermediate object on which we are setting values always behaves as a copy.
```

```
For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.
```

In [44]:

```
# Drop columns with > 30% missing data
threshold = len(data) * 0.3
data = data.dropna(thresh=threshold, axis=1)
```

In [45]:

```
# Forward-fill categorical values
categorical_cols = ['Company Name', 'Industry', 'Sector', 'Type of ownership']
data[categorical_cols] = data[categorical_cols].fillna(method='ffill')
```

```
/tmp/ipython-input-3094392204.py:3: FutureWarning:

DataFrame.fillna with 'method' is deprecated and will raise in a future version. Use obj.ffill() or obj.bfill() instead.
```

Standardizing Data

In [46]:

```
# Extract minimum salary
data['Min Salary'] = data['Salary Estimate'].str.extract(r'(\d+)').astype(float)
```

In [47]:

```
# Extract maximum salary
data['Max Salary'] = data['Salary Estimate'].str.extract(r'-\s*(\d+)').astype(float)
```

In [48]:

```
# Compute average salary
data['Avg Salary'] = (data['Min Salary'] + data['Max Salary'])/ 2
```

In [49]:

```
# Drop old salary column
data.drop('Salary Estimate', axis=1, inplace=True)
```

Feature Engineering

In [50]:

```
# Extract keywords from Job Description
data['Python'] = data['Job Description'].str.contains('Python',case=False, na=False).ast
data['Excel'] = data['Job Description'].str.contains('Excel',case=False, na=False).astyp
```

In [51]:

```
# Create a tech skills score
data['Tech_Skills'] = data['Python'] + data['Excel']
```

Location Splits

In [52]:

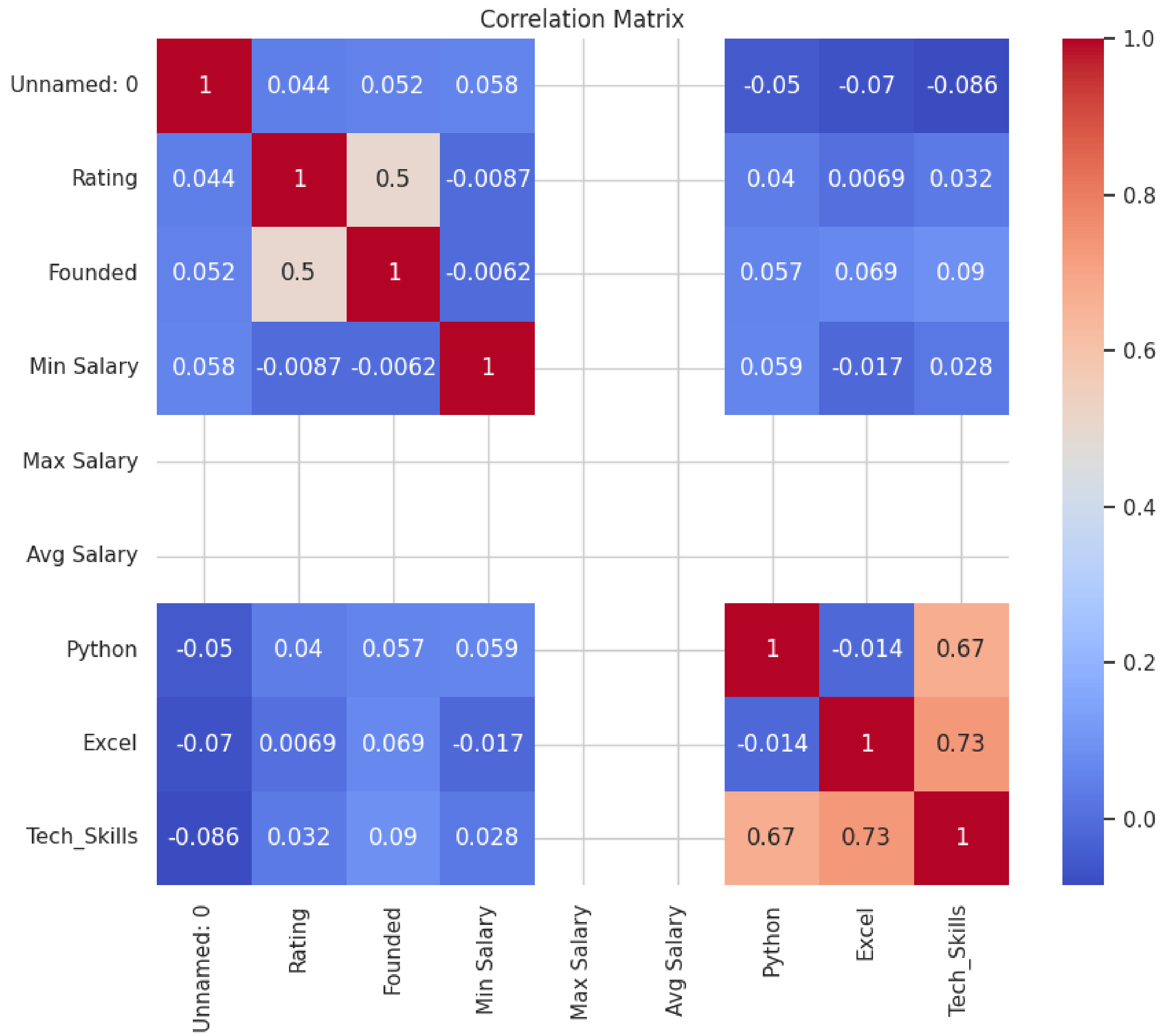
```
# Extract city and state from location
data['City'] = data['Location'].str.split(',', expand=True)[0]
data['State'] = data['Location'].str.split(',', expand=True)[1]
```

Statistics

Analyze relationships using correlation and significance tests.

In [53]:

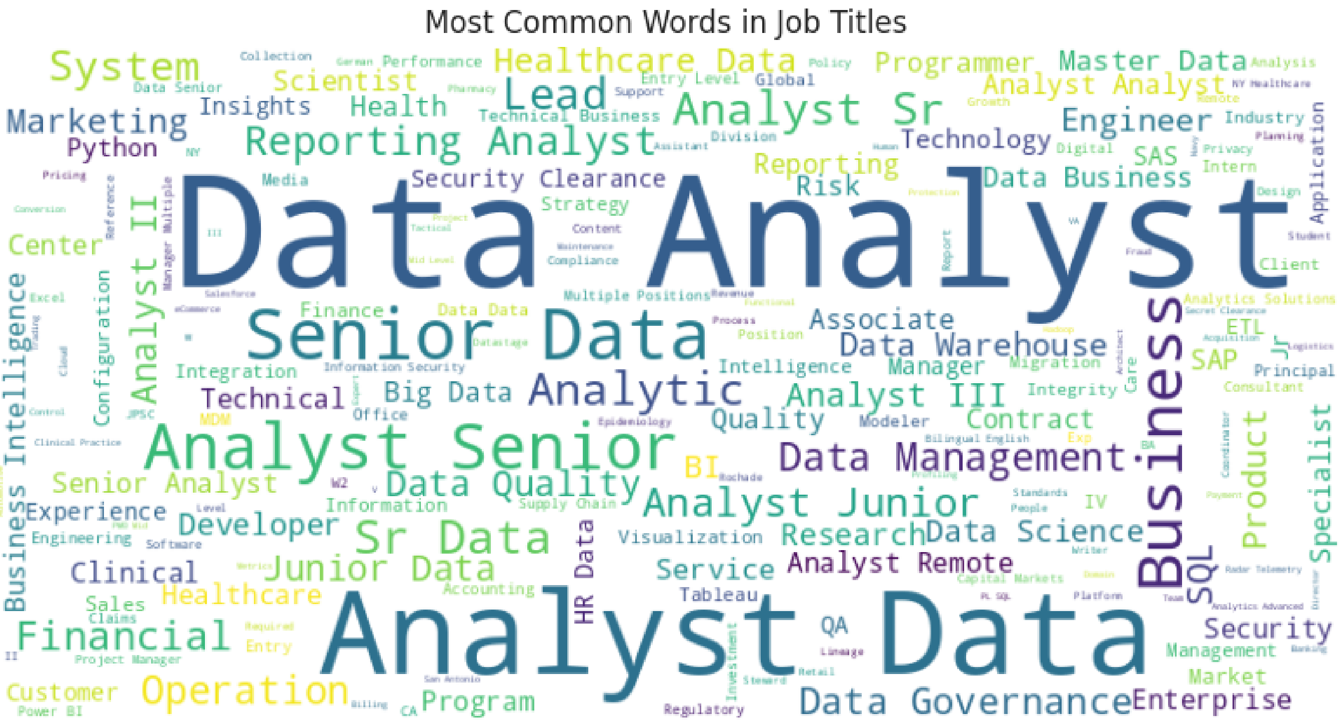
```
# Keep only numeric columns
numeric_data = data.select_dtypes(include=['number'])

# Plot correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(numeric_data.corr(), annot=True, cmap="coolwarm")
plt.title("Correlation Matrix")
plt.show()
```



Word Cloud: Most common words in Job Titles

In [54]:

```python
from wordcloud import WordCloud
text = " ".join(data['Job Title'].astype(str))
wordcloud = WordCloud(width=800, height=400, background_color="white").generate(text)
plt.figure(figsize=(10,6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title("Most Common Words in Job Titles")
plt.show()
```



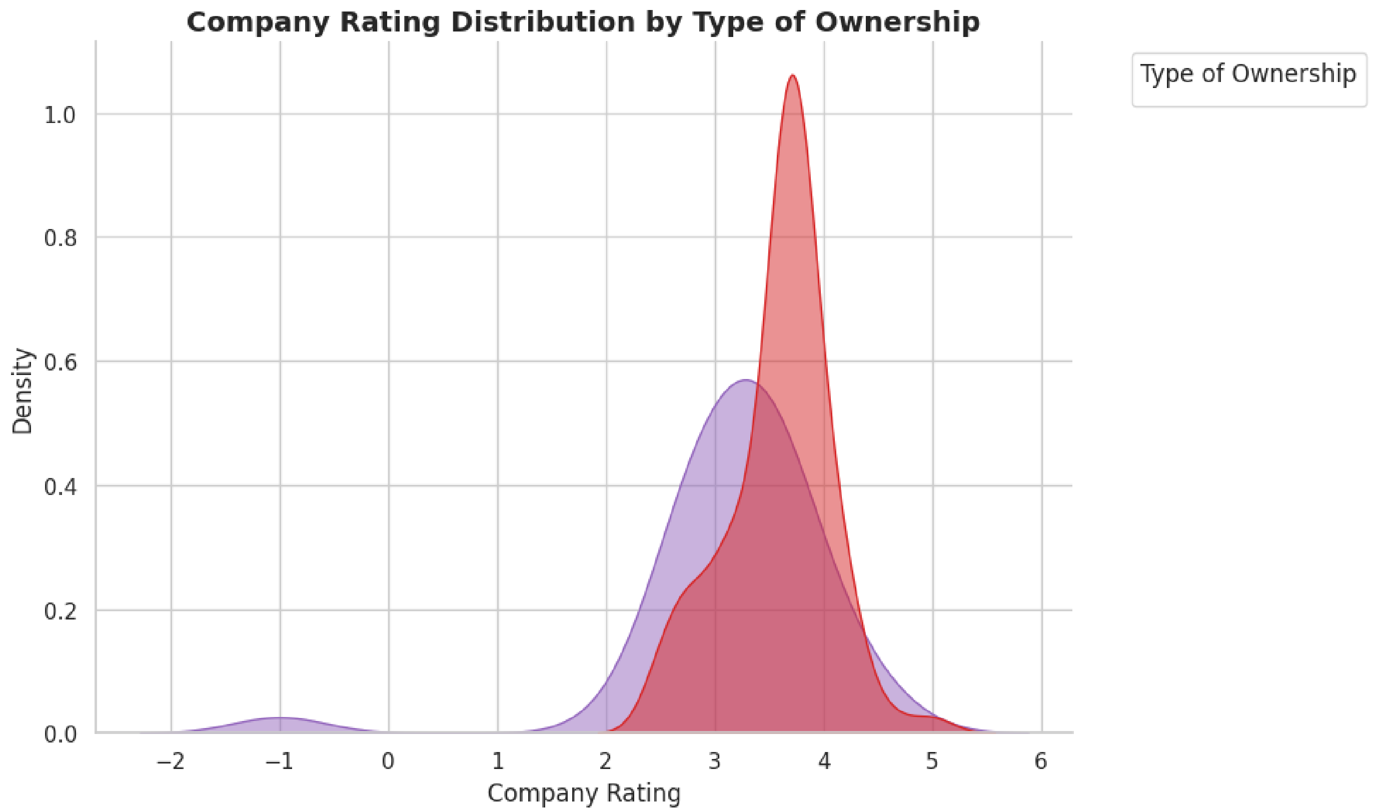
KDE Plot: Company Rating by Type of Ownership (Custom Colors)

In [55]:

```python
plt.figure(figsize=(10,6))
sns.set_theme(style="whitegrid")

# Define custom colors for each ownership type
ownership_colors = {
    'Private': '#1f77b4',      # Blue
    'Public': '#ff7f0e',       # Orange
    'Nonprofit': '#2ca02c',    # Green
    'Subsidiary or Business Segment': '#d62728', # Red
    'Government': '#9467bd',   # Purple
    'Other': '#8c564b'         # Brown
}

# Map only colors that exist in the data
hue_order = [col for col in ownership_colors.keys() if col in data['Type of ownership'].
palette = {k: ownership_colors[k] for k in hue_order}

# KDE Plot
sns.kdeplot(
    data=data,
    x='Rating',
    hue='Type of ownership',
```

```
    fill=True,
    common_norm=False,
    alpha=0.5,
    palette=palette,
    hue_order=hue_order
)

# Labels and title
plt.title("Company Rating Distribution by Type of Ownership", fontsize=14, fontweight='b
plt.xlabel("Company Rating", fontsize=12)
plt.ylabel("Density", fontsize=12)
plt.legend(title='Type of Ownership', bbox_to_anchor=(1.05, 1), loc='upper left')

# Clean style
sns.despine()
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-1431969422.py:34: UserWarning:

No artists with labels found to put in legend.  Note that artists whose label start with
an underscore are ignored when legend() is called with no argument.
```

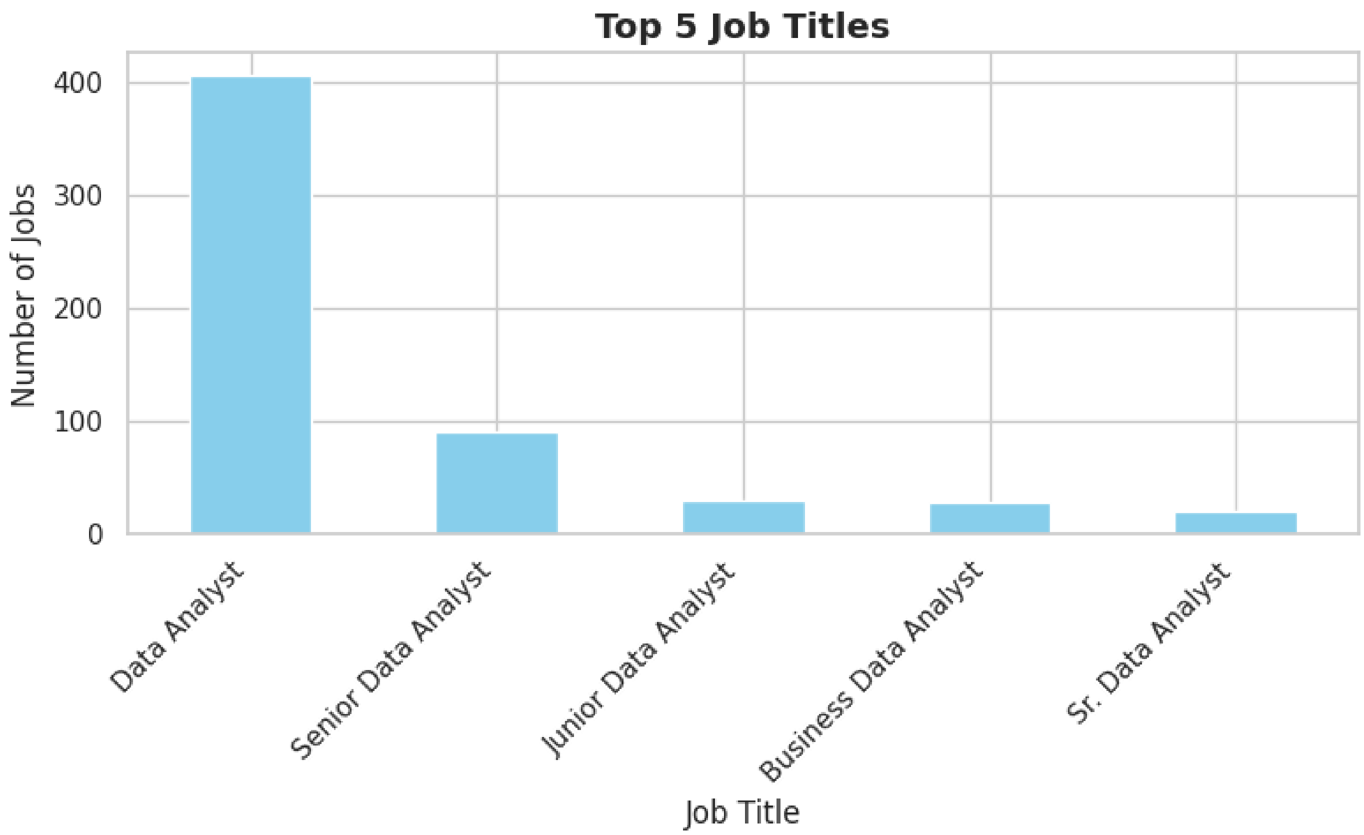


Simple Bar Graph: Top 5 Job Titles

In [56]:

```
top_jobs = data['Job Title'].value_counts().head(5)

plt.figure(figsize=(8,5))
top_jobs.plot(kind='bar', color='skyblue')

plt.title("Top 5 Job Titles", fontsize=14, fontweight='bold')
plt.xlabel("Job Title")
plt.ylabel("Number of Jobs")
```

```
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



Model Development

Data Splitting

Split into features and target:

In [57]:

```
from sklearn.model_selection import train_test_split
# Define features and target
features = ['Rating', 'Tech_Skills', 'Size', 'Founded']
X = data[features]
y = data['Avg Salary']
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42
```

Model Training

In [58]:

```
# ================================
# Model Training for Data Analyst Jobs
# ================================

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
```

```python
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, r2_score
from sklearn.preprocessing import LabelEncoder

# 1. Load Dataset
data = pd.read_csv("DataAnalyst.csv")

# 2. Basic Cleaning
data = data.drop_duplicates()
data = data.dropna(subset=["Salary Estimate", "Rating"])  # remove rows with no salary/r

# 3. Extract Min, Max, and Avg Salary
data['MinSalary'] = data['Salary Estimate'].str.extract(r'(\d+)').astype(float)
data['MaxSalary'] = data['Salary Estimate'].str.extract(r'(\d+)\D*$').astype(float)
data['AvgSalary'] = (data['MinSalary'] + data['MaxSalary']) / 2

# Drop old column
data.drop(['Salary Estimate', 'MinSalary', 'MaxSalary'], axis=1, inplace=True)

# 4. Encode categorical variables
categorical_cols = ['Company Name', 'Location', 'Size', 'Type of ownership', 'Industry',
for col in categorical_cols:
    if col in data.columns:
        le = LabelEncoder()
        data[col] = le.fit_transform(data[col].astype(str))

# 5. Extract skills from Job Description
data['Python'] = data['Job Description'].str.contains('Python', case=False, na=False).as
data['Excel'] = data['Job Description'].str.contains('Excel', case=False, na=False).asty
data['SQL'] = data['Job Description'].str.contains('SQL', case=False, na=False).astype(i
data['Tech_Skills'] = data['Python'] + data['Excel'] + data['SQL']

# 6. Feature selection
features = ['Rating', 'Company Name', 'Location', 'Size', 'Type of ownership', 'Industry
X = data[features]
y = data['AvgSalary']

# 7. Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42

# 8. Model Training
model = RandomForestRegressor(n_estimators=200, random_state=42)
model.fit(X_train, y_train)

# 9. Predictions & Evaluation
y_pred = model.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"MAE: {mae:.2f}")
print(f"R² Score: {r2:.4f}")

# 10. Feature Importance
feat_importance = pd.Series(model.feature_importances_, index=features).sort_values(asce
plt.figure(figsize=(8,5))
sns.barplot(x=feat_importance.values, y=feat_importance.index)
plt.title("Feature Importance in Salary Prediction")
plt.show()
```

```
MAE: 17.59
R² Score: 0.1343
```