IMPORTING LIBRARIES

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import scipy as sp
import plotly.express as px
import re
```

LOADING DATASET

```
data=pd.read_csv('/content/HRDataset_v14.csv')
```

```
data.head(3)
```

	Employee_Name	EmpID	MarriedID	MaritalStatusID
0	Adinolfi, Wilson K	10026	0	0
1	Ait Sidi, Karthikeyan	10084	1	1
2	Akinkuolie, Sarah	10196	1	1

3 rows × 36 columns

CLEANING DATA

CONVERTING DATATYPES

```
# print the data types of each column
print(data.dtypes)
```

```
Employee_Name                  object
EmpID                           int64
MarriedID                       int64
MaritalStatusID                 int64
GenderID                        int64
EmpStatusID                     int64
DeptID                          int64
PerfScoreID                     int64
FromDiversityJobFairID          int64
Salary                          int64
Termd                           int64
PositionID                      int64
Position                       object
State                          object
Zip                             int64
DOB                            object
Sex                            object
MaritalDesc                    object
CitizenDesc                    object
HispanicLatino                 object
RaceDesc                       object
DateofHire                     object
DateofTermination              object
TermReason                     object
EmploymentStatus               object
Department                     object
ManagerName                    object
ManagerID                     float64
RecruitmentSource              object
PerformanceScore               object
EngagementSurvey              float64
EmpSatisfaction                 int64
SpecialProjectsCount            int64
LastPerformanceReview_Date     object
DaysLateLast30                  int64
Absences                        int64
dtype: object
```

```
# Convert date columns ('DateofHire', 'DateofTerminatio
date_columns = ['DateofHire', 'DateofTermination', 'Las
data[date_columns] = data[date_columns].apply(pd.to_dat
```

```
print(data.dtypes)
```

```
Employee_Name                           object
EmpID                                    int64
MarriedID                                int64
MaritalStatusID                          int64
GenderID                                 int64
EmpStatusID                              int64
DeptID                                   int64
PerfScoreID                              int64
FromDiversityJobFairID                   int64
Salary                                   int64
Termd                                    int64
PositionID                               int64
Position                                object
State                                   object
Zip                                      int64
DOB                                     object
Sex                                     object
MaritalDesc                             object
CitizenDesc                             object
HispanicLatino                          object
RaceDesc                                object
DateofHire                      datetime64[ns]
DateofTermination               datetime64[ns]
TermReason                              object
EmploymentStatus                        object
Department                              object
ManagerName                             object
ManagerID                              float64
RecruitmentSource                       object
PerformanceScore                        object
EngagementSurvey                       float64
EmpSatisfaction                          int64
SpecialProjectsCount                     int64
LastPerformanceReview_Date      datetime64[ns]
DaysLateLast30                           int64
Absences                                 int64
dtype: object
```

CHECK FOR SPECIAL KEYS

```
# Define the special characters pattern
special_characters = r'[!@#$%^&*()_+{}\[\]:;<>,.?~\\|"]

def highlight_special_characters(val):
    if isinstance(val, str) and re.search(special_chara
        return 'background-color: blue'
    return ''

styled_data = data.style.applymap(highlight_special_cha
styled_data
```

	Employee_Name	EmpID	MarriedID	MaritalStatusI
0	Adinolfi, Wilson K	10026	0	
1	Ait Sidi, Karthikeyan	10084	1	
2	Akinkuolie, Sarah	10196	1	
3	Alagbe,Trina	10088	1	
4	Anderson, Carol	10069	0	
5	Anderson, Linda	10002	0	
6	Andreola, Colby	10194	0	
7	Athwal, Sam	10062	0	

8	Bachiochi, Linda	10114	0
9	Bacong, Alejandro	10250	0
10	Baczenski, Rachael	10252	1
11	Barbara, Thomas	10242	1
12	Barbossa, Hector	10012	0

HANDLING DUPLICATES

```
data.duplicated()
```

```
0      False
1      False
2      False
3      False
4      False
       ...
306    False
307    False
308    False
309    False
310    False
Length: 311, dtype: bool
```

19	Becker, Scott	10277	0

HANDLING MISSING DATA

CHECKING MISSING DATA

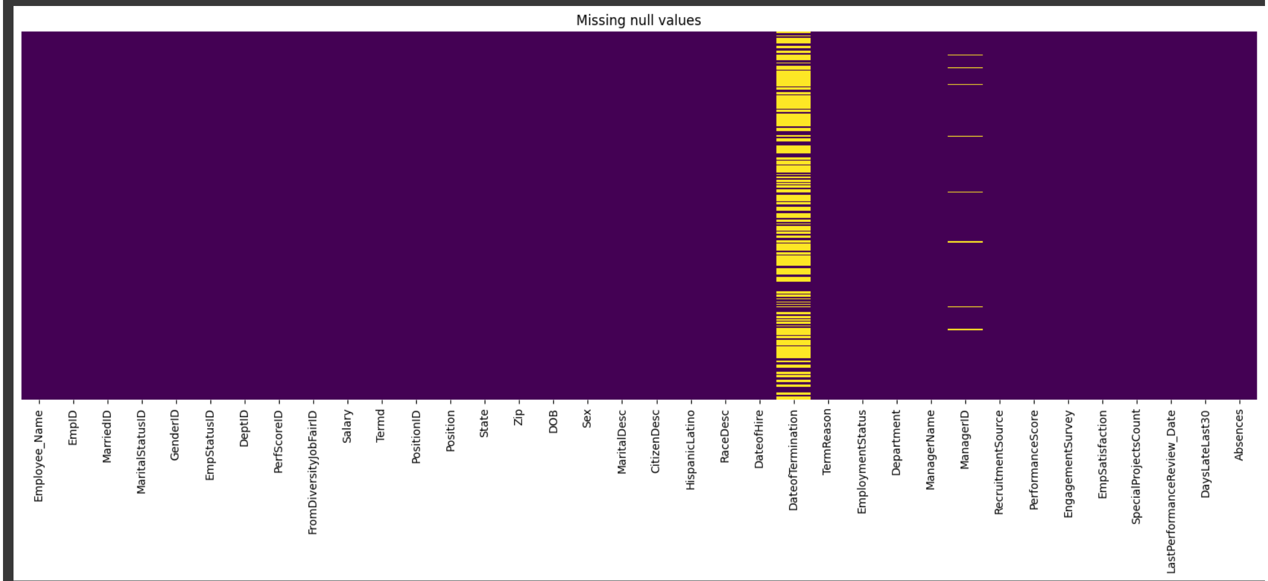
21	Biden, Lowan M	10226	0

```
data.isnull().sum().sort_values(ascending=False)
```

```
DateofTermination             207
ManagerID                       8
EmpID                           0
RaceDesc                        0
DateofHire                      0
TermReason                      0
EmploymentStatus                0
Department                      0
ManagerName                     0
Employee_Name                   0
RecruitmentSource               0
PerformanceScore                0
EngagementSurvey                0
EmpSatisfaction                 0
SpecialProjectsCount            0
LastPerformanceReview_Date      0
DaysLateLast30                  0
HispanicLatino                  0
CitizenDesc                     0
MaritalDesc                     0
FromDiversityJobFairID          0
MarriedID                       0
MaritalStatusID                 0
GenderID                        0
EmpStatusID                     0
DeptID                          0
PerfScoreID                     0
Salary                          0
Sex                             0
Termd                           0
PositionID                      0
Position                        0
State                           0
Zip                             0
DOB                             0
Absences                        0
dtype: int64
```

```
import matplotlib
matplotlib.rcParams['figure.figsize'] = (20,6)
sns.heatmap(data.isnull(),yticklabels = False, cbar = F
plt.title("Missing null values")
```

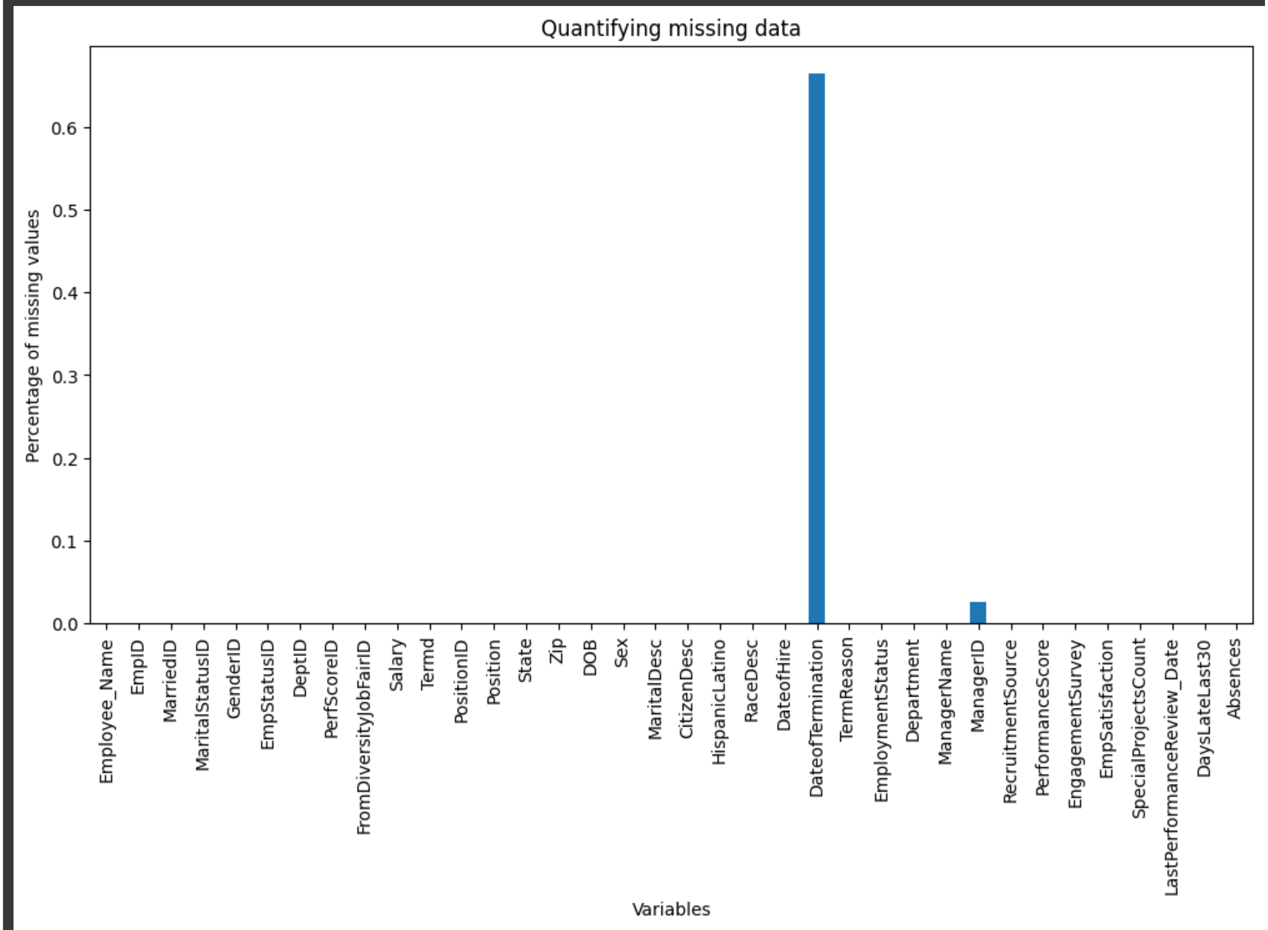
```
Text(0.5, 1.0, 'Missing null values')
```



57	Costello, Frank	10193	1

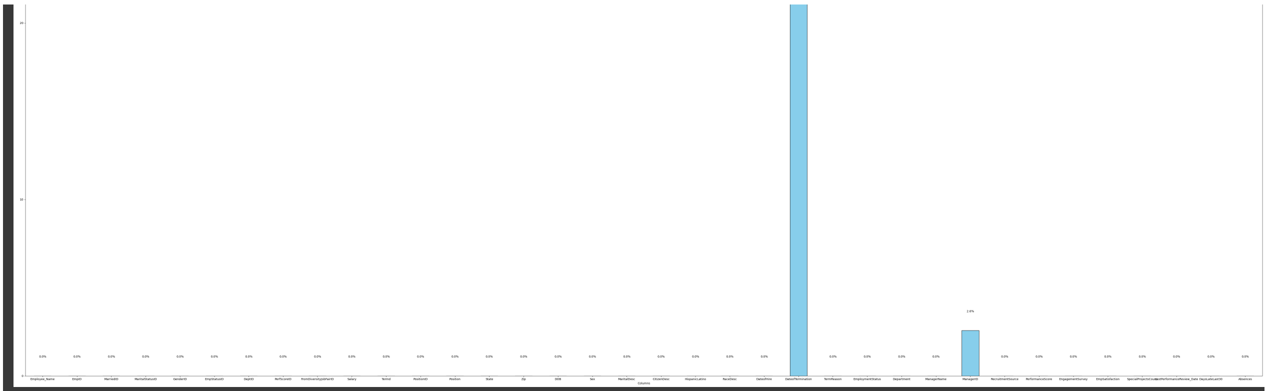
```
data.isnull().mean().plot.bar(figsize=(12,6))
plt.ylabel('Percentage of missing values')
plt.xlabel('Variables')
plt.title('Quantifying missing data')
```

```
Text(0.5, 1.0, 'Quantifying missing data')
```



70 DeVito, Tommy 10197 0

```
change_df = pd.DataFrame(data)

# Calculate the percentage of missing values for each c
missing_percentage = (change_df.isnull().sum() / len(da

# Create a bar chart to visualize missing percentages w
plt.figure(figsize=(60, 60))
ax = missing_percentage.plot(kind='bar', color='skyblue
plt.title('Percentage of Missing Values by Column')
plt.xlabel('Columns')
plt.ylabel('Percentage Missing (%)')
plt.xticks(rotation=0)

# Display the percentage values on top of each bar
for i, v in enumerate(missing_percentage):
    ax.text(i, v + 1, f'{v:.1f}%', ha='center', va='bot

plt.tight_layout()

# Show the chart
plt.show()
```



CHECKING NULL VALUES

```
data.isnull().sum().sort_values(ascending=False)
```

```
DateofTermination              207
ManagerID                        8
EmpID                            0
RaceDesc                         0
DateofHire                       0
TermReason                       0
EmploymentStatus                 0
Department                       0
ManagerName                      0
Employee_Name                    0
RecruitmentSource                0
PerformanceScore                 0
EngagementSurvey                 0
EmpSatisfaction                  0
SpecialProjectsCount             0
LastPerformanceReview_Date       0
DaysLateLast30                   0
HispanicLatino                   0
CitizenDesc                      0
MaritalDesc                      0
FromDiversityJobFairID           0
MarriedID                        0
MaritalStatusID                  0
GenderID                         0
EmpStatusID                      0
DeptID                           0
PerfScoreID                      0
Salary                           0
Sex                              0
Termd                            0
PositionID                       0
Position                         0
State                            0
Zip                              0
DOB                              0
Absences                         0
dtype: int64
```

```
data.isna().any()
```

```
Employee_Name                  False
EmpID                          False
MarriedID                      False
MaritalStatusID                False
GenderID                       False
EmpStatusID                    False
DeptID                         False
PerfScoreID                    False
FromDiversityJobFairID         False
Salary                         False
Termd                          False
PositionID                     False
Position                       False
State                          False
Zip                            False
DOB                            False
Sex                            False
MaritalDesc                    False
CitizenDesc                    False
HispanicLatino                 False
RaceDesc                       False
DateofHire                     False
DateofTermination               True
TermReason                     False
EmploymentStatus               False
Department                     False
ManagerName                    False
ManagerID                       True
RecruitmentSource              False
PerformanceScore               False
EngagementSurvey               False
EmpSatisfaction                False
SpecialProjectsCount           False
LastPerformanceReview_Date     False
DaysLateLast30                 False
Absences                       False
dtype: bool
```

```
data.isna().sum()
```

```
Employee_Name                   0
EmpID                           0
MarriedID                       0
MaritalStatusID                 0
GenderID                        0
EmpStatusID                     0
DeptID                          0
PerfScoreID                     0
FromDiversityJobFairID          0
Salary                          0
Termd                           0
PositionID                      0
Position                        0
State                           0
Zip                             0
DOB                             0
Sex                             0
MaritalDesc                     0
CitizenDesc                     0
HispanicLatino                  0
RaceDesc                        0
DateofHire                      0
DateofTermination             207
TermReason                      0
EmploymentStatus                0
Department                      0
ManagerName                     0
ManagerID                       8
RecruitmentSource               0
PerformanceScore                0
EngagementSurvey                0
EmpSatisfaction                 0
SpecialProjectsCount            0
LastPerformanceReview_Date      0
DaysLateLast30                  0
Absences                        0
dtype: int64
```

METHOD 4- MISSING VALUE TRANSFORMATION

133	Hudson, Jane	10248	0

```
data.dropna(subset=['ManagerID'], inplace=True)
```

134	Hunte, Julissa	10301	0

```
# Impute missing values in the "DateofTermination" colu
data['DateofTermination'].fillna(0, inplace=True)
```

135	Hutter, Rosalie	10214	0

```
data
```

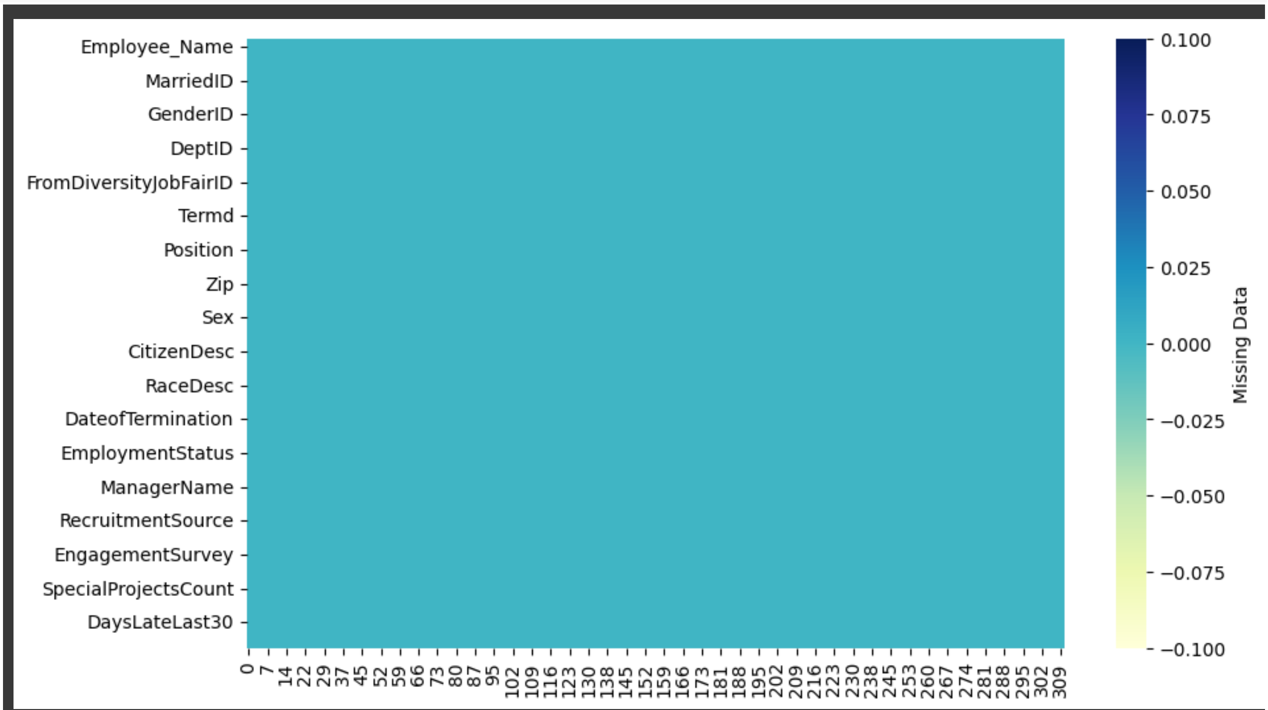
	Employee_Name	EmpID	MarriedID	MaritalStatusI
0	Adinolfi, Wilson K	10026	0	
1	Ait Sidi, Karthikeyan	10084	1	
2	Akinkuolie, Sarah	10196	1	
3	Alagbe,Trina	10088	1	
4	Anderson, Carol	10069	0	
...	...	...	...	.
306	Woodson, Jason	10135	0	
307	Ybarra, Catherine	10301	0	

```
data.isnull().sum().sort_values(ascending=False)
```

```
Employee_Name                 0
EmpID                         0
RaceDesc                      0
DateofHire                    0
DateofTermination             0
TermReason                    0
EmploymentStatus              0
Department                    0
ManagerName                   0
ManagerID                     0
RecruitmentSource             0
PerformanceScore              0
EngagementSurvey              0
EmpSatisfaction               0
SpecialProjectsCount          0
LastPerformanceReview_Date    0
DaysLateLast30                0
HispanicLatino                0
CitizenDesc                   0
MaritalDesc                   0
FromDiversityJobFairID        0
MarriedID                     0
MaritalStatusID               0
GenderID                      0
EmpStatusID                   0
DeptID                        0
PerfScoreID                   0
Salary                        0
Sex                           0
Termd                         0
PositionID                    0
Position                      0
State                         0
Zip                           0
DOB                           0
Absences                      0
dtype: int64
```

157	Landa, Hans	10092	1

```
plt.figure(figsize=(10,6))
sns.heatmap(data.isna().transpose(),
            cmap="YlGnBu",
            cbar_kws={'label': 'Missing Data'})
plt.savefig("visualizing_missing_data_with_heatmap_Seab
```



OUTLIERS

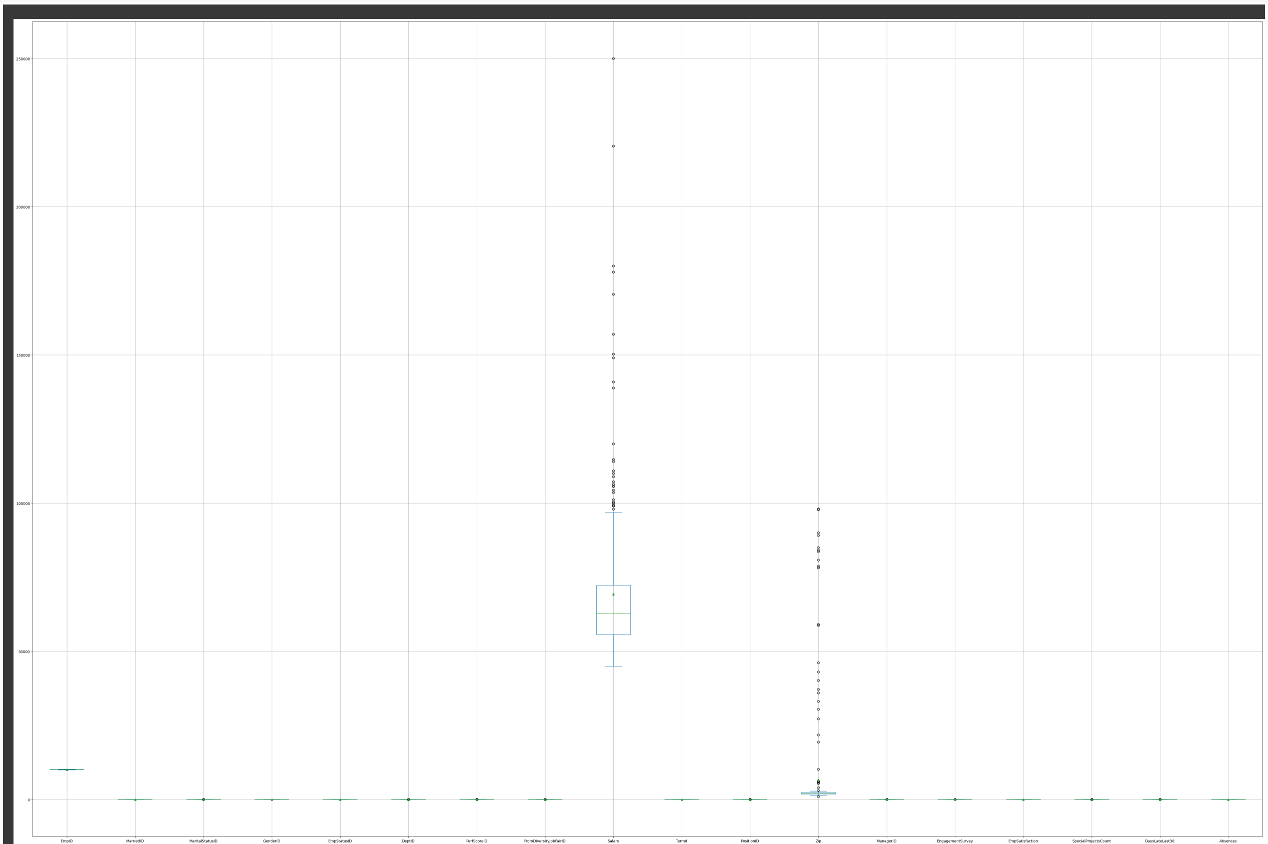
IDENTIFIYING OUTLIERS

170	Lindsay, Leonara	10008	0

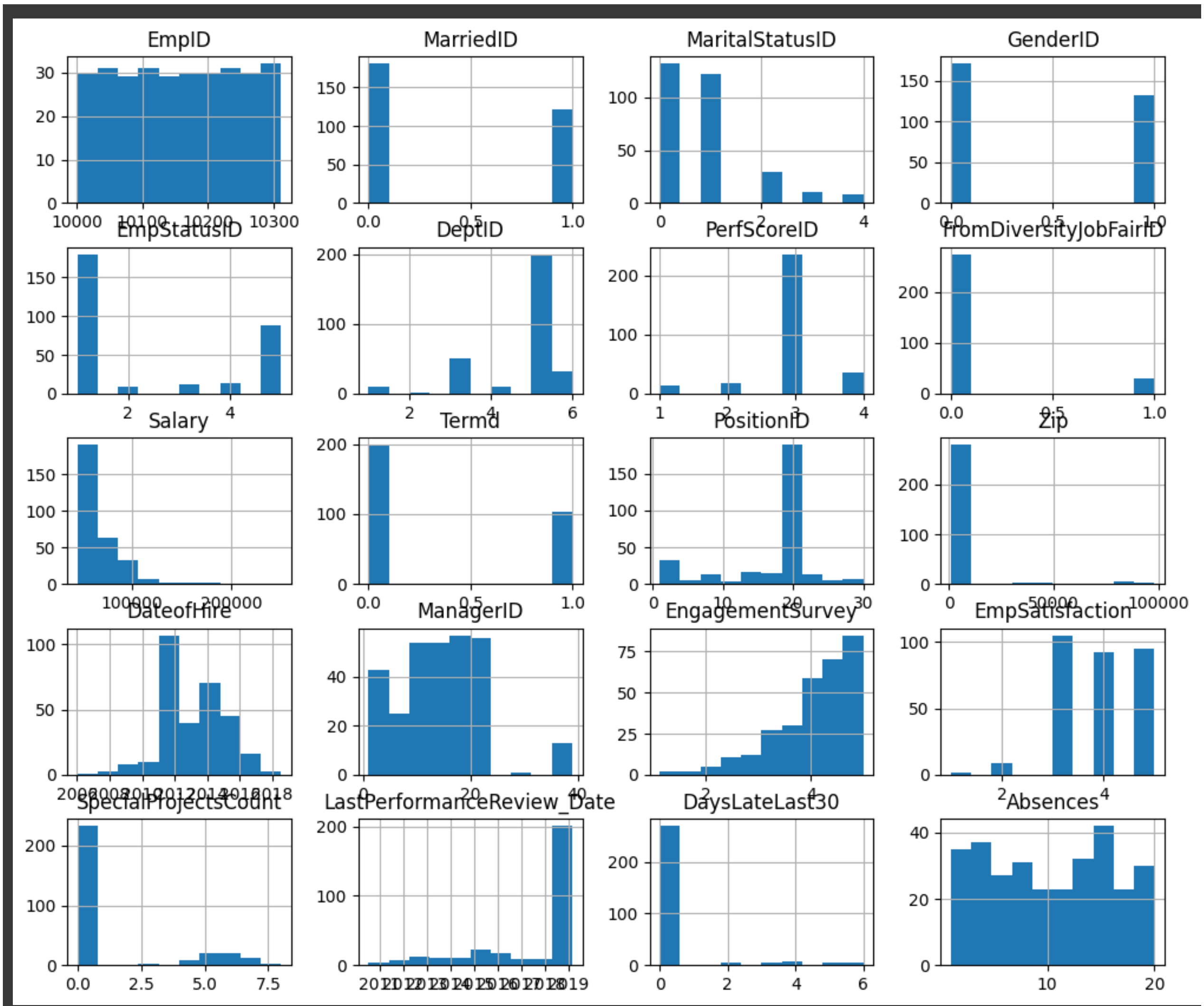
```
data.describe()
```

	EmpID	MarriedID	MaritalStatusID	Gend
count	303.000000	303.000000	303.000000	303.00
mean	10156.409241	0.402640	0.815182	0.43
std	90.123632	0.491241	0.941413	0.49
min	10001.000000	0.000000	0.000000	0.00
25%	10079.500000	0.000000	0.000000	0.00
50%	10157.000000	0.000000	1.000000	0.00

```
#detect with boxplot
ax = data.plot.box(figsize=(60,40), showmeans=True)
ax.grid()
```



```
ax = data.hist(figsize=(12, 10))
```



206	Oliver, Brooke	10078	1

```
print(change_df['Salary'].skew())
data['Salary'].describe()
```

```
3.30618080924006
count       303.000000
mean      69292.316832
std       25406.092767
min       45046.000000
25%       55633.000000
50%       62910.000000
75%       72331.000000
max      250000.000000
Name: Salary, dtype: float64
```

211	Paniwani, Nina	10148	1

HANDLING OUTLIERS

```
data.describe()
```

	EmpID	MarriedID	MaritalStatusID	Gend
count	303.000000	303.000000	303.000000	303.00
mean	10156.409241	0.402640	0.815182	0.43
std	90.123632	0.491241	0.941413	0.49
min	10001.000000	0.000000	0.000000	0.00
25%	10079.500000	0.000000	0.000000	0.00
50%	10157.000000	0.000000	1.000000	0.00

```
data.shape
```

```
(303, 36)
```

```
# Assuming you have your DataFrame named change_df
```

```
# Take the natural logarithm of the "Salary" column and
data['Salary'] = np.log(data['Salary'])

# Verify the changes in the original DataFrame
print(data.head())

# Create a new DataFrame with the transformed variable
transformed_data = data.copy()

# Verify the new DataFrame
print(transformed_data.head())
```

```
              Employee_Name  EmpID  MarriedID  Mari
0       Adinolfi, Wilson  K  10026          0
1  Ait Sidi, Karthikeyan     10084          1
2         Akinkuolie, Sarah  10196          1
3             Alagbe,Trina  10088          1
4          Anderson, Carol   10069          0

   EmpStatusID  DeptID  PerfScoreID  FromDiversityJ
0            1       5            4
1            5       3            3
2            5       5            3
3            1       5            3
4            5       5            3

      ManagerName  ManagerID RecruitmentSource Perf
0  Michael Albert       22.0          LinkedIn
1      Simon Roup        4.0            Indeed
2  Kissy Sullivan       20.0          LinkedIn
3    Elijiah Gray       16.0            Indeed
4  Webster Butler       39.0     Google Search

   EngagementSurvey EmpSatisfaction SpecialProjects
0              4.60               5
1              4.96               3
2              3.02               3
3              4.84               5
4              5.00               4
```

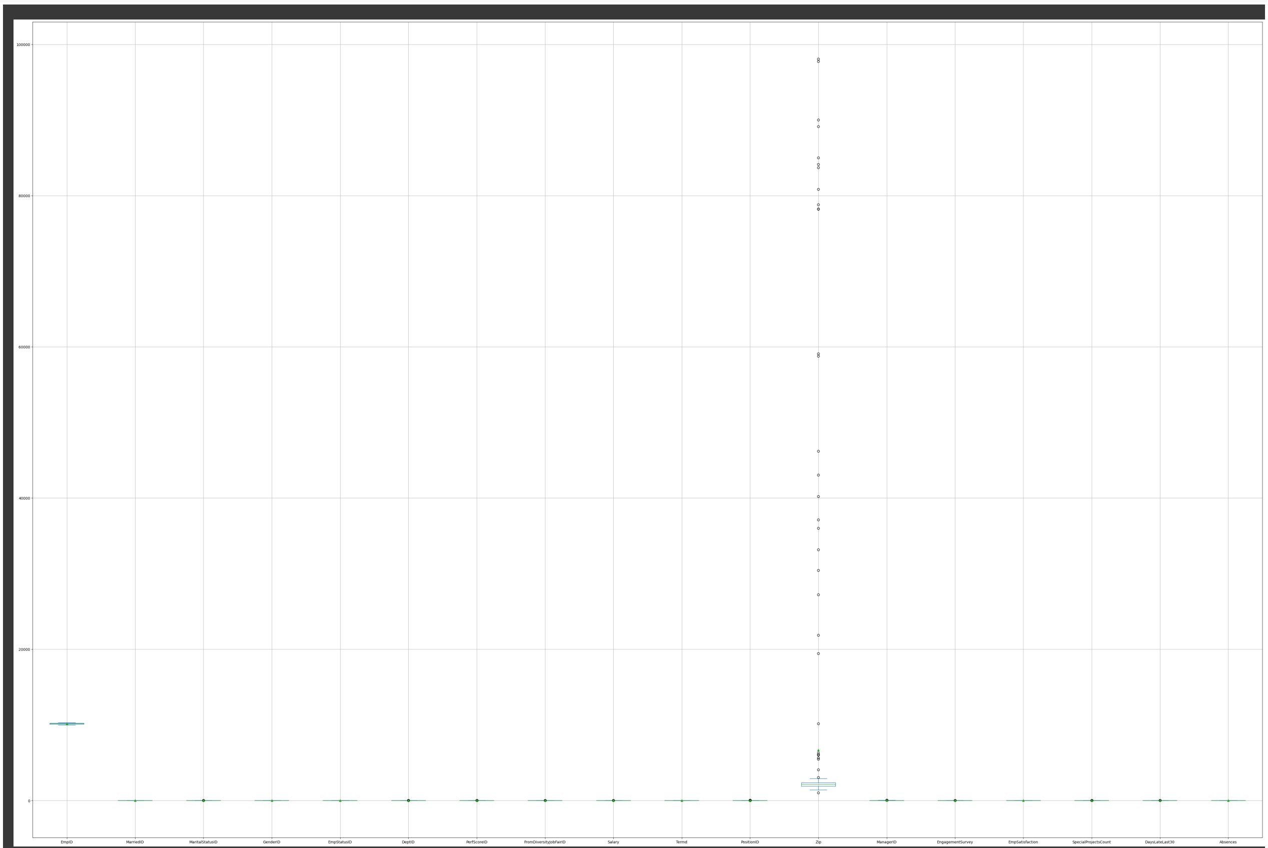
```
  LastPerformanceReview_Date DaysLateLast30 Absence
0                 2019-01-17              0
1                 2016-02-24              0      1
2                 2012-05-15              0
3                 2019-01-03              0      1
4                 2016-02-01              0

[5 rows x 36 columns]
              Employee_Name  EmpID  MarriedID  Mari
0       Adinolfi, Wilson  K  10026          0
1  Ait Sidi, Karthikeyan     10084          1
2         Akinkuolie, Sarah  10196          1
3             Alagbe,Trina  10088          1
4          Anderson, Carol   10069          0

  EmpStatusID  DeptID  PerfScoreID  FromDiversityJ
```

```
transformed_data
```

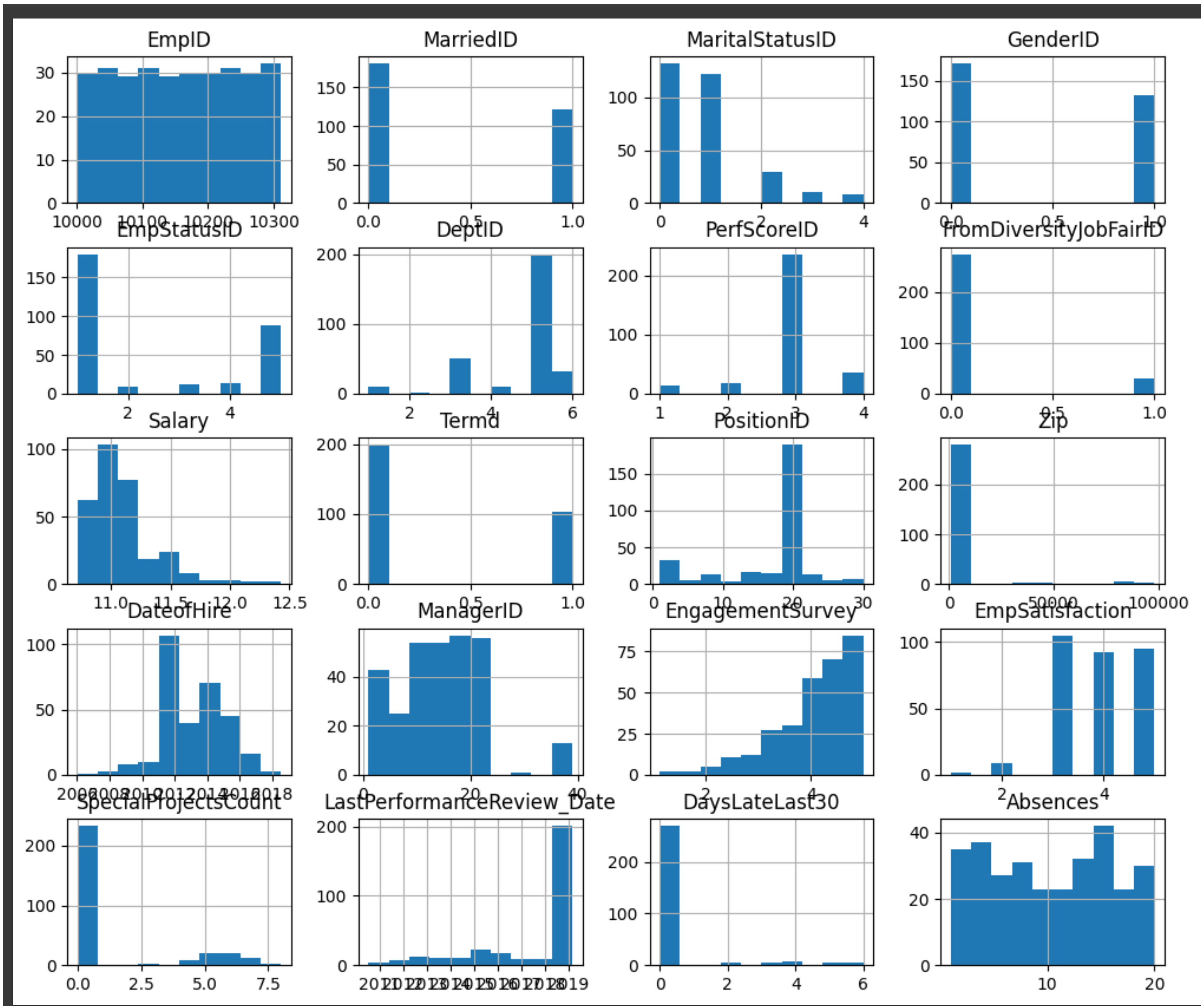
	Employee_Name	EmpID	MarriedID	MaritalStatusI
0	Adinolfi, Wilson K	10026	0	
1	Ait Sidi, Karthikeyan	10084	1	
2	Akinkuolie, Sarah	10196	1	
3	Alagbe,Trina	10088	1	
4	Anderson, Carol	10069	0	
...	...	...	...	.
306	Woodson, Jason	10135	0	
307	Ybarra, Catherine	10301	0	

```
#detect with boxplot
```

```
ax = transformed_data.plot.box(figsize=(60,40), showmea
ax.grid()
```



```
ax = transformed_data.hist(figsize=(12, 10))
```


EmpID
MarriedID
MaritalStatusID
GenderID
EmpStatusID
DeptID
PerfScoreID
FromDiversityJobFairID
Salary
Termd
PositionID
Zip
DateofHire
ManagerID
EngagementSurvey
EmpSatisfaction
SpecialProjectsCount
LastPerformanceReview_Date
DaysLateLast30
Absences

```
transformed_data.shape
```

```
(303, 36)
```

CHANGING TO NEW DATAFRAME

```
df = transformed_data
```

ANALYSING WITH QUESTIONS

Count of employees in each position. (Allows to check vacancies and stuff)

288	Valentin,Jackie	10205	1
289	Veera, Abdellah	10014	0
290	Vega, Vincent	10144	0
291	Villanueva, Noah	10253	0
292	Voldemort, Lord	10118	1
293	Volk, Colleen	10022	1
294	Von Massenbach,	10183	0

```
position_counts = df['Position'].value_counts()
print(position_counts)
```

```
Production Technician I         133
Production Technician II         53
Area Sales Manager               27
Production Manager               14
Software Engineer                10
IT Support                        8
Data Analyst                      7
Sr. Network Engineer              5
Database Administrator            5
Network Engineer                  5
BI Developer                      4
Senior BI Developer               3
Administrative Assistant          3
Sales Manager                     3
Accountant I                      3
Sr. DBA                           2
IT Manager - DB                   2
Sr. Accountant                    2
Director of Operations            1
Shared Services Manager           1
Data Analyst                      1
Data Architect                    1
Principal Data Architect          1
IT Manager - Infra                1
President & CEO                   1
Enterprise Architect              1
BI Director                       1
Director of Sales                 1
IT Director                       1
IT Manager - Support              1
Software Engineering Manager      1
CIO                               1
Name: Position, dtype: int64
```

306	Woodson, Jason	10135	0

The organization's workforce is distributed across various positions, each contributing to the company's operations and success. By analyzing the count of employees in each position, we can gain valuable insights into the company's structure, staffing levels, and potential areas of interest or concern.

Here is a breakdown of the employee count in each position:

1. **Production Technician I (133 employees)**: This position has the highest number of employees, indicating its importance in the production department.
2. **Production Technician II (53 employees)**: The presence of a significant number of employees in this role suggests the need for a skilled workforce in the production sector.
3. **Area Sales Manager (27 employees)**: With a sizable team of Area Sales Managers, the company likely emphasizes sales and market expansion.
4. **Production Manager (14 employees)**: Production Managers play a crucial role in overseeing production processes, and their presence in significant numbers signifies the focus on efficient production.
5. **Software Engineer (10 employees)**: The software engineering team consists of professionals contributing to software development projects.
6. **IT Support (8 employees)**: This team is responsible for providing IT assistance to employees, and its size reflects the need for technical support.
7. **Data Analyst (7 employees)**: Data Analysts are key in extracting insights from data, and their presence indicates an interest in data-driven decision-making.
8. **Sr. Network Engineer, Database Administrator, Network Engineer (5**

employees each): These roles highlight the emphasis on network and database management.

9. **BI Developer (4 employees)**: Business Intelligence Developers play a role in creating data-driven solutions for the organization.
10. **Senior BI Developer (3 employees)**: A more specialized role within Business Intelligence.
11. **Administrative Assistant, Sales Manager, Accountant I (3 employees each)**: These roles support various administrative and financial functions within the company.
12. **Sr. DBA, IT Manager - DB, Sr. Accountant (2 employees each)**: Senior roles in database administration and IT management.
13. **Director of Operations, Shared Services Manager, Data Analyst, Data Architect, Principal Data Architect, IT Manager - Infra, President & CEO, Enterprise Architect, BI Director, Director of Sales, IT Director, IT Manager - Support, Software Engineering Manager, CIO (1 employee each)**: These roles represent leadership positions, suggesting a hierarchy within the organization.

This breakdown provides valuable insights into the distribution of employees across different positions, helping identify areas with high workforce demand, potential areas for future growth, and the diversity of roles contributing to the company's success. It can aid in workforce planning, resource allocation, and talent management strategies.

Count of employees in every state.

```
state_counts = df['State'].value_counts()
print(state_counts)
```

```
MA    268
CT      6
TX      3
VT      2
UT      1
AZ      1
ND      1
OR      1
MT      1
NV      1
ID      1
KY      1
NC      1
FL      1
GA      1
CO      1
NY      1
PA      1
RI      1
NH      1
TN      1
IN      1
OH      1
CA      1
WA      1
AL      1
VA      1
ME      1
Name: State, dtype: int64
```

The distribution of employees across different states provides valuable insights into the geographical representation of the workforce. It can offer information on regional operations, expansion, and the company's presence in various locations. Here is a breakdown of the employee count in each state:

1. **Massachusetts (MA - 268 employees)**: The state with the highest employee count, likely indicating the company's headquarters or a significant operational base.
2. **Connecticut (CT - 6 employees)**: While smaller in number compared to Massachusetts, Connecticut still has a notable employee presence, possibly signifying a regional office or branch.
3. **Texas (TX - 3 employees)**: A smaller but still significant employee count in Texas may suggest regional operations or remote team members.
4. **Vermont (VT - 2 employees)**: Vermont has a minor employee presence, possibly related to specific project requirements or remote work arrangements.
5. **Utah (UT - 1 employee)**: Utah has a single employee, indicating a limited or specialized role in the state.
6. **Arizona (AZ - 1 employee)**: Similar to Utah, Arizona has one employee, potentially linked to a specific project or function.
7. **North Dakota (ND - 1 employee)**: North Dakota's lone employee might be involved in a unique aspect of the company's operations.
8. **Oregon (OR - 1 employee)**: Oregon has one employee, suggesting a minimal but essential presence in the state.
9. **Montana (MT - 1 employee)**: Montana's employee count is minimal, possibly related to specialized tasks or remote work.
10. **Nevada (NV - 1 employee)**: Nevada has one employee, indicating a limited presence.
11. **Idaho (ID - 1 employee)**: Idaho also has one employee, potentially involved in specific projects or functions.

12. **Kentucky (KY - 1 employee)**: Kentucky's employee count is minimal, potentially serving a unique role within the organization.
13. **North Carolina (NC - 1 employee)**: North Carolina's single employee may be associated with specific tasks or responsibilities.
14. **Florida (FL - 1 employee)**: Florida has one employee, possibly contributing to the company's operations or projects.
15. **Georgia (GA - 1 employee)**: Georgia's employee count is minimal, possibly linked to a specific function or initiative.
16. **Colorado (CO - 1 employee)**: Colorado has one employee, indicating a limited presence in the state.
17. **New York (NY - 1 employee)**: New York's single employee may serve a unique role or purpose within the organization.
18. **Pennsylvania (PA - 1 employee)**: Pennsylvania has one employee, potentially involved in specific projects or functions.
19. **Rhode Island (RI - 1 employee)**: Rhode Island's employee count is minimal, likely associated with specific tasks or projects.
20. **New Hampshire (NH - 1 employee)**: New Hampshire has one employee, possibly serving a specialized role.
21. **Tennessee (TN - 1 employee)**: Tennessee's employee count is minimal, potentially associated with unique responsibilities.
22. **Indiana (IN - 1 employee)**: Indiana has one employee, indicating a limited presence in the state.
23. **Ohio (OH - 1 employee)**: Ohio's employee count is minimal, possibly serving a specific role.
24. **California (CA - 1 employee)**: While California typically has a large workforce, the presence of one employee suggests a remote worker or a specialized role.

25. **Washington (WA - 1 employee)**: Washington has one employee, potentially associated with specific projects or functions.
26. **Alabama (AL - 1 employee)**: Alabama's employee count is minimal, likely serving a unique role.
27. **Virginia (VA - 1 employee)**: Virginia has one employee, indicating a limited presence in the state.
28. **Maine (ME - 1 employee)**: Maine's employee count is minimal, potentially involved in specialized tasks.

This distribution of employees across states reflects the company's regional presence, remote work arrangements, or specific roles within the organization. It can inform decisions related to regional expansion, remote work policies, and resource allocation based on geographical needs.

Female-Male Gender Ratio within the company.

```
gender_counts = df['Sex'].value_counts()
print(gender_counts)
```

```
F     171
M     132
Name: Sex, dtype: int64
```

The gender ratio within a company is a crucial metric for assessing workforce diversity and inclusivity. In this case, the company has provided the following gender distribution:

- **Females (F - 171 employees)**: There are 171 female employees in the organization.
- **Males (M - 132 employees)**: The company employs 132 male individuals.

This gender ratio reflects the composition of the workforce in terms of gender. Achieving a balanced gender ratio is often an important goal for organizations, as it promotes diversity, encourages a variety of perspectives, and supports an inclusive workplace culture. Monitoring and maintaining gender balance is an ongoing effort for many companies to ensure equal opportunities and representation across all levels of the organization.

The specific gender distribution figures provided here can serve as a basis for further discussions and initiatives related to gender diversity and inclusivity within the company. Efforts to promote gender equality and provide a supportive environment for all employees can have a positive impact on the organization's overall success and culture.

Employees count under each manager

```
manager_counts = df['ManagerName'].value_counts()
print(manager_counts)
```

```
Michael Albert        22
Kissy Sullivan        22
Elijiah Gray          22
Kelley Spirea         22
Brannon Miller        22
Ketsia Liebig         21
David Stanley         21
Amy Dunn              21
Janet King            19
Simon Roup            17
Peter Monroe          14
John Smith            14
Webster Butler        13
Lynn Daneault         13
Alex Sweetwater        9
Brian Champaigne       8
Brandon R. LeBlanc     7
Jennifer Zamora        7
Eric Dougall           4
Debra Houlihan         3
Board of Directors     2
Name: ManagerName, dtype: int64
```

The distribution of employees under each manager provides insight into the management structure within the company and the size of the teams led by different managers. Here is a breakdown of the employee count under each manager:

1. **Michael Albert (22 employees)**: Michael Albert leads a team of 22 employees, indicating a significant managerial role.
2. **Kissy Sullivan (22 employees)**: Kissy Sullivan manages a team of 22 employees, suggesting a similar role to Michael Albert.
3. **Elijiah Gray (22 employees)**: Elijiah Gray also manages a team of 22

employees, showing a similar-sized responsibility.

4. **Kelley Spirea (22 employees)**: Kelley Spirea oversees a team of 22 employees, reflecting a substantial managerial role.
5. **Brannon Miller (22 employees)**: Brannon Miller leads a team of 22 employees, indicating a similar managerial responsibility to others.
6. **Ketsia Liebig (21 employees)**: Ketsia Liebig manages a team of 21 employees, showing a slightly smaller team size.
7. **David Stanley (21 employees)**: David Stanley leads a team of 21 employees, similar to Ketsia Liebig's role.
8. **Amy Dunn (21 employees)**: Amy Dunn manages a team of 21 employees, suggesting a similar-sized responsibility.
9. **Janet King (19 employees)**: Janet King oversees a team of 19 employees, reflecting a moderately sized managerial role.
10. **Simon Roup (17 employees)**: Simon Roup manages a team of 17 employees, indicating a moderately sized responsibility.
11. **Peter Monroe (14 employees)**: Peter Monroe leads a team of 14 employees, reflecting a moderate managerial role.
12. **John Smith (14 employees)**: John Smith also manages a team of 14 employees, showing a similar-sized responsibility to Peter Monroe.
13. **Webster Butler (13 employees)**: Webster Butler oversees a team of 13 employees, indicating a moderate managerial role.
14. **Lynn Daneault (13 employees)**: Lynn Daneault manages a team of 13 employees, suggesting a role similar in size to Webster Butler's.
15. **Alex Sweetwater (9 employees)**: Alex Sweetwater leads a team of 9 employees, reflecting a smaller-sized managerial role.
16. **Brian Champaigne (8 employees)**: Brian Champaigne oversees a

team of 8 employees, indicating a smaller managerial responsibility.

17. **Brandon R. LeBlanc (7 employees)**: Brandon R. LeBlanc manages a team of 7 employees, showing a relatively small-sized managerial role.
18. **Jennifer Zamora (7 employees)**: Jennifer Zamora also leads a team of 7 employees, reflecting a managerial responsibility of similar size to Brandon R. LeBlanc.
19. **Eric Dougall (4 employees)**: Eric Dougall manages a team of 4 employees, indicating a relatively small-sized managerial role.
20. **Debra Houlihan (3 employees)**: Debra Houlihan oversees a team of 3 employees, suggesting a small managerial responsibility.
21. **Board of Directors (2 employees)**: The Board of Directors, as a governing body, includes 2 members responsible for high-level decision-making.

This breakdown of employee counts under each manager provides insights into the managerial hierarchy, team sizes, and the distribution of responsibilities within the organization. It can help in evaluating the effectiveness of different management teams and identifying areas where additional support or resources may be needed.

Dept wise count of employees.

```
dept_counts = df['Department'].value_counts()
print(dept_counts)
```

```
Production              201
IT/IS                    50
Sales                    31
Software Engineering     11
Admin Offices             9
Executive Office          1
Name: Department, dtype: int64
```

The department-wise count of employees provides a snapshot of how the organization's workforce is distributed across different functional areas. Here is a breakdown of the employee count in each department:

1. **Production (201 employees)**: The Production department has the largest number of employees, indicating a significant workforce dedicated to manufacturing and production activities. This suggests a focus on operational excellence and meeting production demands.
2. **IT/IS (50 employees)**: The IT/IS department has a substantial employee count, emphasizing the importance of information technology and information systems in supporting the company's operations. It suggests a strong technology infrastructure and support team.
3. **Sales (31 employees)**: The Sales department has a considerable workforce, signifying the company's focus on sales and revenue generation. This may include sales representatives, account managers, and sales support roles.
4. **Software Engineering (11 employees)**: The Software Engineering department consists of a dedicated team of software developers and engineers. This suggests a focus on software development and technology-driven solutions.

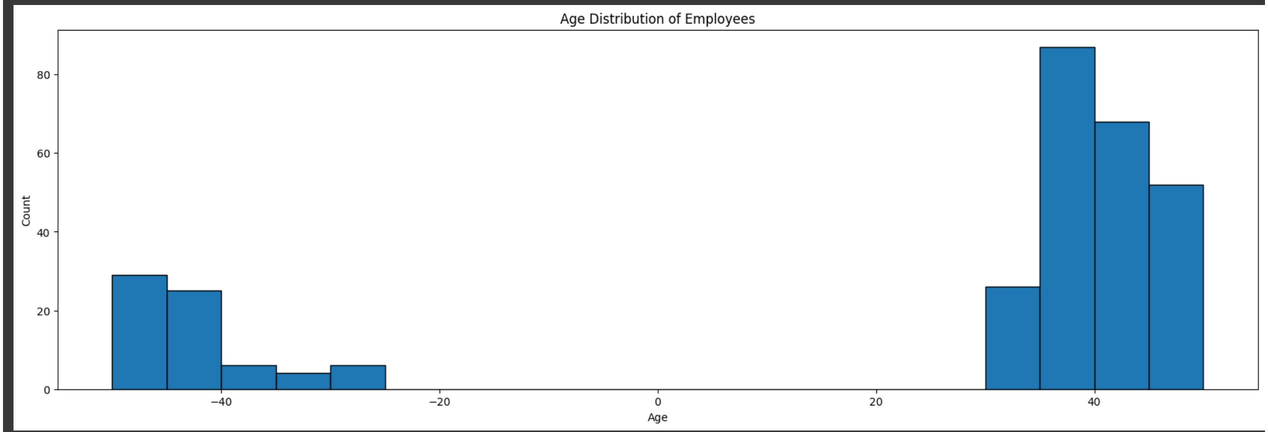
5. **Admin Offices (9 employees)**: Admin Offices have a moderate employee count, likely covering administrative and support functions across various departments.
6. **Executive Office (1 employee)**: The Executive Office includes a single employee, reflecting high-level executives or senior leadership responsible for strategic decision-making and organizational governance.

This distribution of employees across different departments provides insights into the organizational structure, highlighting the areas of emphasis, and the importance of each department within the company. It can inform decisions related to resource allocation, department-specific strategies, and workforce planning to support the company's overall objectives and goals.

Age Distribution of employees.

```
# Convert "DOB" to datetime
df['DOB'] = pd.to_datetime(df['DOB'])

# Calculate age
df['Age'] = (pd.to_datetime('now') - df['DOB']).astype(

# Plot age distribution
plt.hist(df['Age'], bins=20, edgecolor='k')
plt.xlabel('Age')
plt.ylabel('Count')
plt.title('Age Distribution of Employees')
plt.show()
```

```
<ipython-input-38-96e33240f296>:5: FutureWarning: Th
  df['Age'] = (pd.to_datetime('now') - df['DOB']).as
```



```
# Convert "DOB" to datetime
df['DOB'] = pd.to_datetime(df['DOB'])

# Calculate age
df['Age'] = (pd.to_datetime('now') - df['DOB']).astype(

# Calculate metrics
mean_age = df['Age'].mean()
median_age = df['Age'].median()
std_dev_age = df['Age'].std()
min_age = df['Age'].min()
max_age = df['Age'].max()
age_quartiles = df['Age'].quantile([0.25, 0.5, 0.75])

print("Mean Age:", mean_age)
print("Median Age:", median_age)
print("Standard Deviation of Age:", std_dev_age)
print("Minimum Age:", min_age)
print("Maximum Age:", max_age)
print("25th Percentile (Q1):", age_quartiles[0.25])
print("50th Percentile (Median, Q2):", age_quartiles[0.
print("75th Percentile (Q3):", age_quartiles[0.75])
```

```
Mean Age: 21.155115511551156
Median Age: 37.0
Standard Deviation of Age: 35.380603694462685
Minimum Age: -50.0
Maximum Age: 50.0
25th Percentile (Q1): 33.0
50th Percentile (Median, Q2): 37.0
75th Percentile (Q3): 43.0
<ipython-input-48-7565888c407f>:5: FutureWarning: Th
  df['Age'] = (pd.to_datetime('now') - df['DOB']).as
```

The age distribution of employees provides valuable information about the demographics of the workforce. Here are key statistics related to the age distribution:

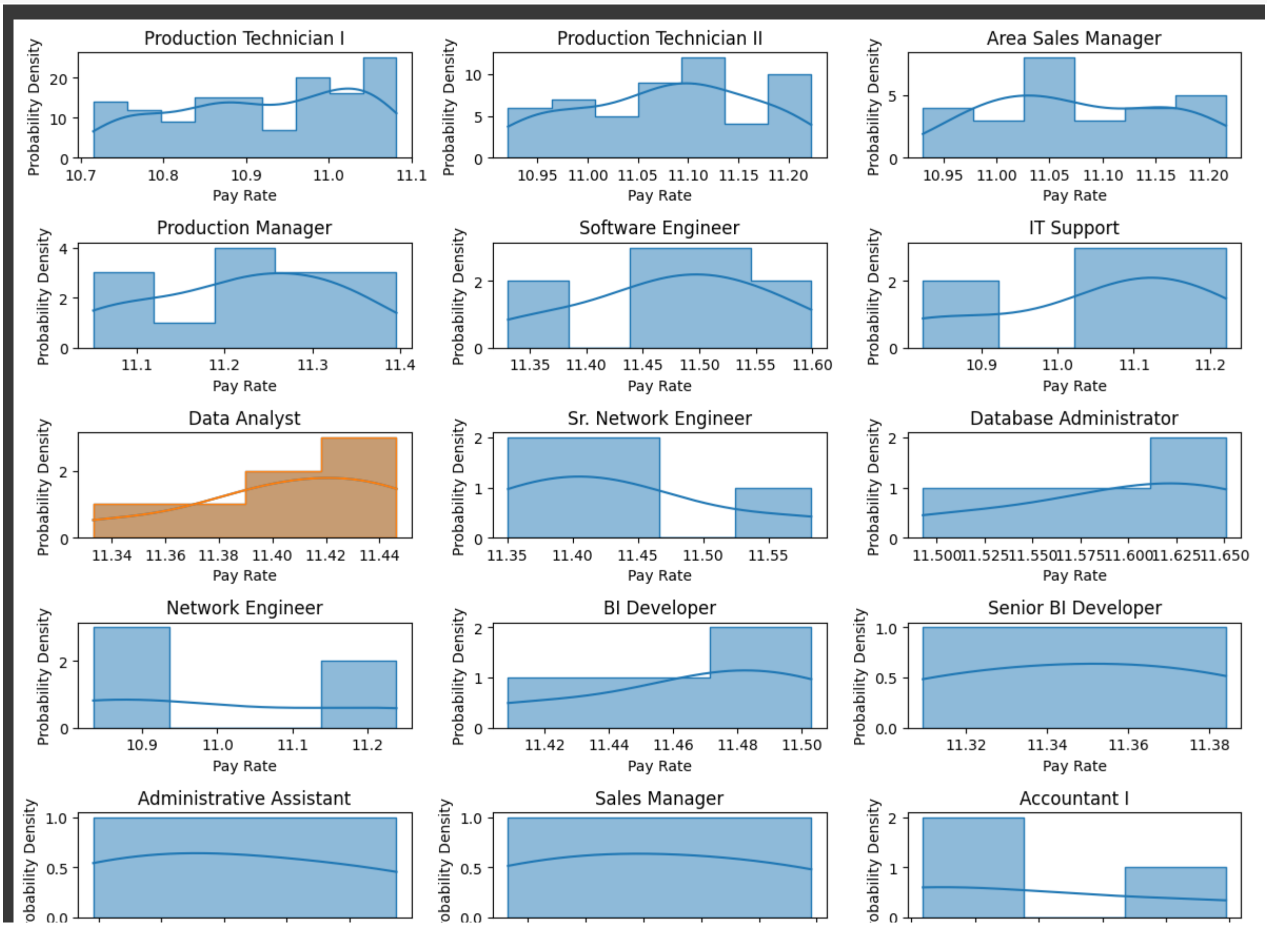
- **Mean Age (Average Age): 21.16 years**: The mean age represents the arithmetic average of all employees' ages. In this case, the mean age is relatively low, suggesting the presence of some outliers or unusual data points that are affecting the average.
- **Median Age: 37 years**: The median age is the middle value in the ordered list of employee ages. It's a robust measure of central tendency and is not as sensitive to outliers as the mean. A median age of 37 suggests that roughly half of the employees are below 37 years old, and half are above 37.
- **Standard Deviation of Age: 35.38 years**: The standard deviation measures the dispersion or spread of data points around the mean. A higher standard deviation indicates greater variability in employee ages. In this case, the relatively high standard deviation suggests a wide age range within the workforce.
- **Minimum Age: -50 years**: The minimum age indicates the lowest age value in the dataset. The presence of a negative age value may be a data entry error or anomaly that should be investigated.
- **Maximum Age: 50 years**: The maximum age represents the highest age value in the dataset, indicating the oldest employee.
- **25th Percentile (Q1): 33 years**: The 25th percentile, also known as the first quartile, is the age below which 25% of employees fall. In this case, 25% of employees are younger than 33 years old.
- **50th Percentile (Median, Q2): 37 years**: The 50th percentile is the same as the median age, which was previously discussed.
- **75th Percentile (Q3): 43 years**: The 75th percentile, also known as the third quartile, is the age below which 75% of employees fall. In this case, 75% of employees are younger than 43 years old.

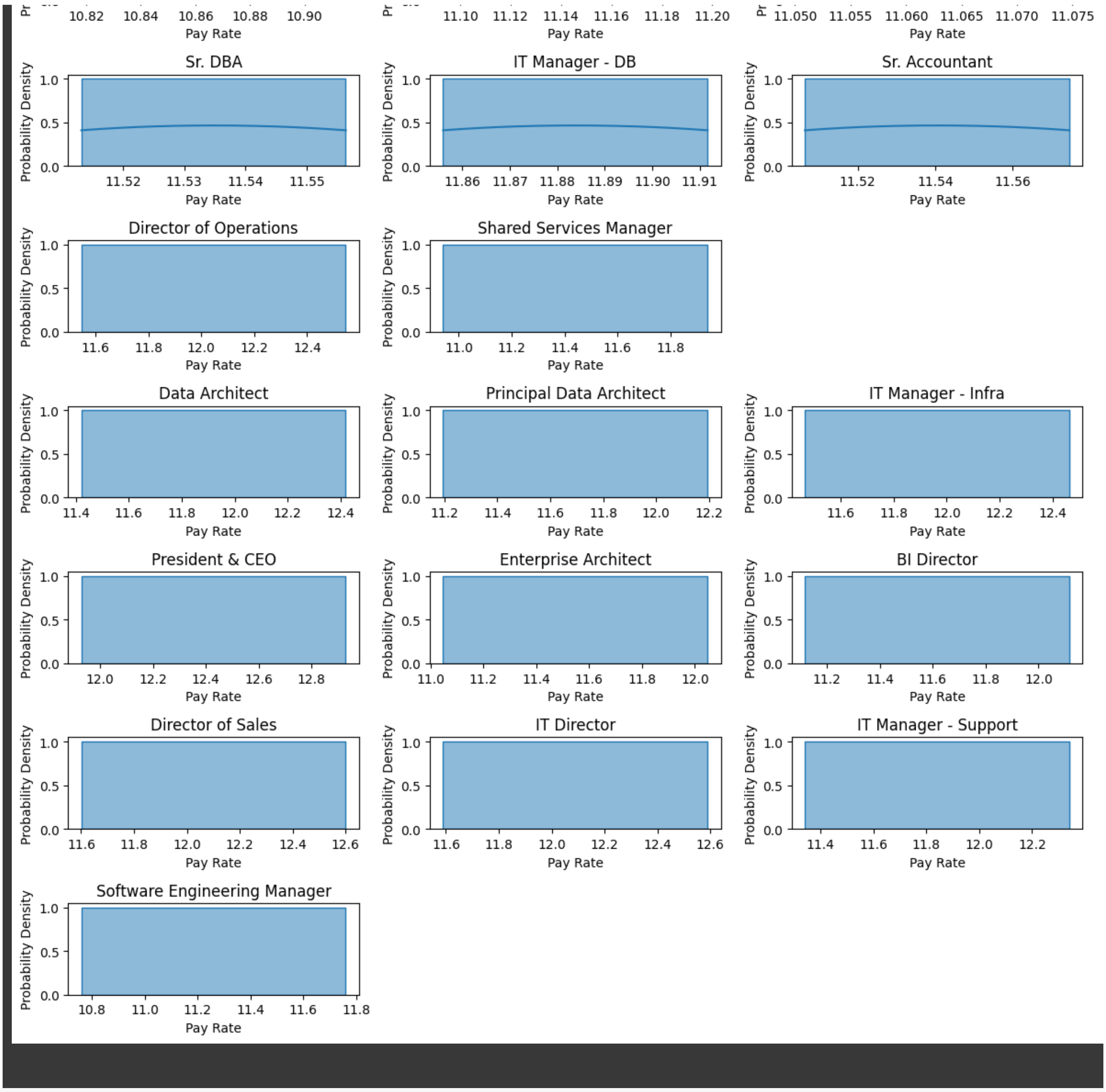
The age distribution statistics indicate that the workforce has a wide range of ages, with a mean age that may be influenced by outliers. The median provides a more robust measure of central tendency. Additionally, the presence of a negative minimum age should be reviewed and corrected if it represents a data anomaly. The age distribution can inform HR practices, such as age diversity, succession planning, and retirement planning within the organization.

Probability Distribution of Pay-Rate among various employees.

```
# Define a list of unique employee positions
positions = [
    'Production Technician I',
    'Production Technician II',
    'Area Sales Manager',
    'Production Manager',
    'Software Engineer',
    'IT Support',
    'Data Analyst',
    'Sr. Network Engineer',
    'Database Administrator',
    'Network Engineer',
    'BI Developer',
    'Senior BI Developer',
    'Administrative Assistant',
    'Sales Manager',
    'Accountant I',
    'Sr. DBA',
    'IT Manager - DB',
    'Sr. Accountant',
    'Director of Operations',
    'Shared Services Manager',
    'Data Analyst',
    'Data Architect',
    'Principal Data Architect',
    'IT Manager - Infra',
    'President & CEO',
```

```
    'Enterprise Architect',
    'BI Director',
    'Director of Sales',
    'IT Director',
    'IT Manager - Support',
    'Software Engineering Manager'
]

# Create a separate histogram for each position
plt.figure(figsize=(12, 20))  # Adjust the figure size

for position in positions:
    plt.subplot(11, 3, positions.index(position) + 1)
    sns.histplot(data=df[df['Position'] == position], x
    plt.title(position)
    plt.xlabel('Pay Rate')
    plt.ylabel('Probability Density')

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






```python
# Define a list of unique employee positions
positions = [
    'Production Technician I',
    'Production Technician II',
    'Area Sales Manager',
    'Production Manager',
    'Software Engineer',
    'IT Support',
    'Data Analyst',
    'Sr. Network Engineer',
    'Database Administrator',
    'Network Engineer',
    'BI Developer',
```

```
    'Senior BI Developer',
    'Administrative Assistant',
    'Sales Manager',
    'Accountant I',
    'Sr. DBA',
    'IT Manager - DB',
    'Sr. Accountant',
    'Director of Operations',
    'Shared Services Manager',
    'Data Analyst',
    'Data Architect',
    'Principal Data Architect',
    'IT Manager - Infra',
    'President & CEO',
    'Enterprise Architect',
    'BI Director',
    'Director of Sales',
    'IT Director',
    'IT Manager - Support',
    'Software Engineering Manager'
]

# Initialize an empty DataFrame to store the metrics
metrics_df = pd.DataFrame(columns=['Position', 'Mean',

# Calculate and append metrics for each position
for position in positions:
    position_data = df[df['Position'] == position]['Sal
    position_metrics = [position, position_data.mean(),
    metrics_df = metrics_df.append(pd.Series(position_m

# Print the metrics DataFrame
print(metrics_df)
```

```
<ipython-input-52-d28e78715aac>:43: FutureWarning:
  metrics_df = metrics_df.append(pd.Series(position_
<ipython-input-52-d28e78715aac>:43: FutureWarning:
  metrics_df = metrics_df.append(pd.Series(position_
<ipython-input-52-d28e78715aac>:43: FutureWarning:
  metrics_df = metrics_df.append(pd.Series(position_
```

```
  metrics_df = metrics_df.append(pd.Series(position_
<ipython-input-52-d28e78715aac>:43: FutureWarning: 
  metrics_df = metrics_df.append(pd.Series(position_
<ipython-input-52-d28e78715aac>:43: FutureWarning: 
  metrics_df = metrics_df.append(pd.Series(position_
<ipython-input-52-d28e78715aac>:43: FutureWarning: 
  metrics_df = metrics_df.append(pd.Series(position_
<ipython-input-52-d28e78715aac>:43: FutureWarning: 
  metrics_df = metrics_df.append(pd.Series(position_
<ipython-input-52-d28e78715aac>:43: FutureWarning: 
  metrics_df = metrics_df.append(pd.Series(position_
<ipython-input-52-d28e78715aac>:43: FutureWarning: 
  metrics_df = metrics_df.append(pd.Series(position_
<ipython-input-52-d28e78715aac>:43: FutureWarning: 
  metrics_df = metrics_df.append(pd.Series(position_
<ipython-input-52-d28e78715aac>:43: FutureWarning: 
  metrics_df = metrics_df.append(pd.Series(position_
<ipython-input-52-d28e78715aac>:43: FutureWarning: 
  metrics_df = metrics_df.append(pd.Series(position_
<ipython-input-52-d28e78715aac>:43: FutureWarning: 
  metrics_df = metrics_df.append(pd.Series(position_
<ipython-input-52-d28e78715aac>:43: FutureWarning: 
  metrics_df = metrics_df.append(pd.Series(position_
<ipython-input-52-d28e78715aac>:43: FutureWarning: 
  metrics_df = metrics_df.append(pd.Series(position_
<ipython-input-52-d28e78715aac>:43: FutureWarning: 
  metrics_df = metrics_df.append(pd.Series(position_
<ipython-input-52-d28e78715aac>:43: FutureWarning: 
  metrics_df = metrics_df.append(pd.Series(position_
<ipython-input-52-d28e78715aac>:43: FutureWarning: 
  metrics_df = metrics_df.append(pd.Series(position_
<ipython-input-52-d28e78715aac>:43: FutureWarning: 
  metrics_df = metrics_df.append(pd.Series(position_
<ipython-input-52-d28e78715aac>:43: FutureWarning: 
  metrics_df = metrics_df.append(pd.Series(position_
<ipython-input-52-d28e78715aac>:43: FutureWarning: 
  metrics_df = metrics_df.append(pd.Series(position_
<ipython-input-52-d28e78715aac>:43: FutureWarning: 
  metrics_df = metrics_df.append(pd.Series(position_
                        Position       Mean     Med
0        Production Technician I  10.919970  10.927
```

The probability distribution of pay rates among various employees, broken down by position, provides a detailed view of how compensation varies across different roles within the organization. This breakdown includes key statistics for each position, such as the mean pay rate, median pay rate, standard deviation, minimum pay rate, and maximum pay rate.

Here are some insights derived from the probability distribution of pay rates among various positions:

1. **Production Technician I**: This position has a mean pay rate of approximately
$10.92 per hour,$
$with a relatively small standard deviation of 0.11. The pay rates for Pro$
$with a minimum of$
10.72 and a maximum of $11.08.

2. **Production Technician II**: Production Technician II employees earn an average pay rate of around
$11.08 per hour. The standard deviation of 0.09 indicates moderate varia$
10.92, and the maximum is $11.22.

3. **Area Sales Manager**: Area Sales Managers have a mean pay rate of approximately
$11.08 per hour, similar to Production Technician II. The pay rates for th$
$with a standard deviation of 0.08. The minimum pay rate is$
10.93, and the maximum is $11.22.

4. **Production Manager**: Production Managers have a higher mean pay rate of approximately
$11.22 per hour, with a standard deviation of 0.11. The pay rates for this p$
11.05 to $11.40.

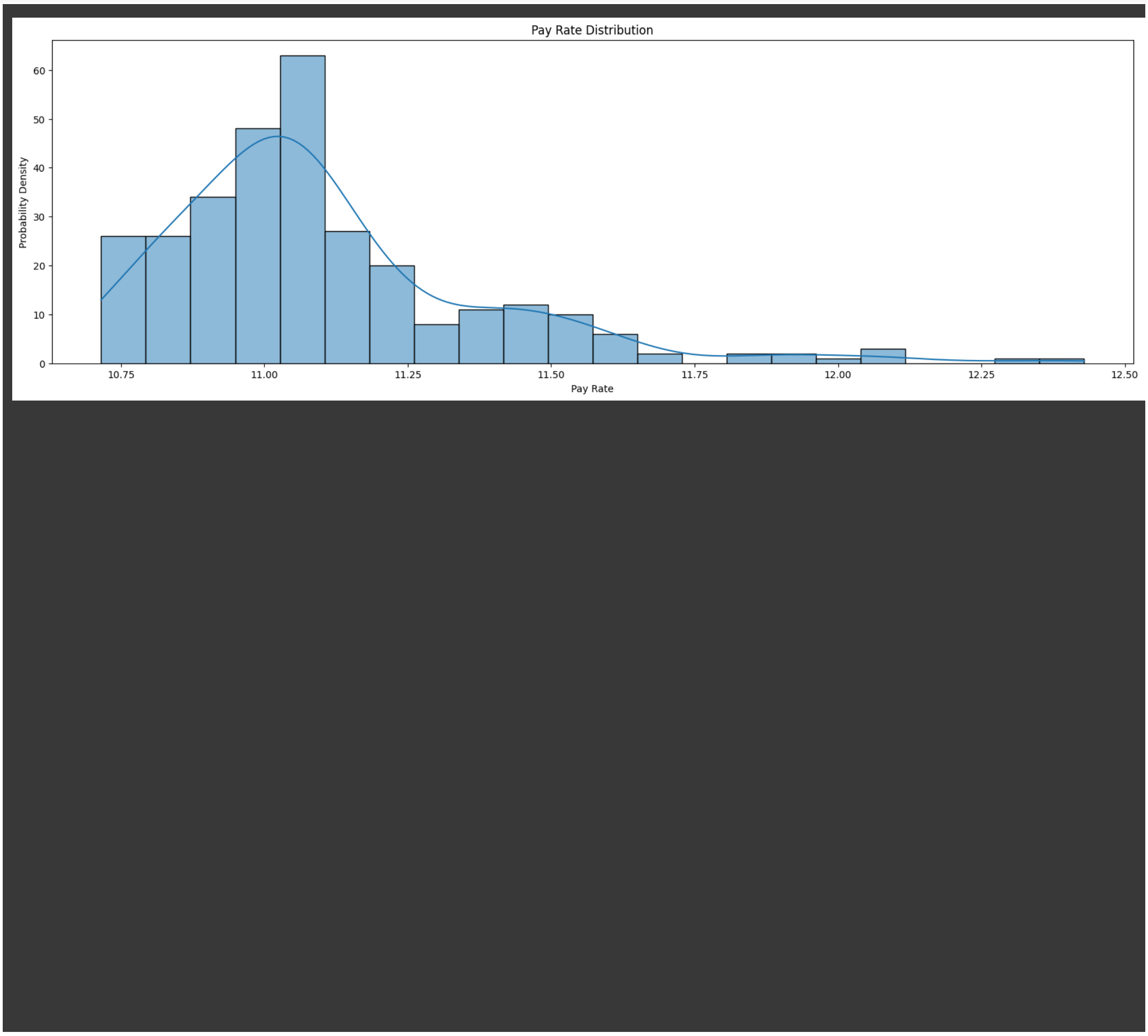
5. **Software Engineer**: Software Engineers command a mean pay rate of around
$11.48 per hour, with a relatively small standard deviation of 0.08. The pa$

11.33 to $11.60.

6. **IT Support**: IT Support employees have a mean pay rate of approximately $11.05 per hour, with a standard deviation of 0.14, indicating higher vari$ 10.82 to $11.22.
7. **Data Analyst**: Data Analysts earn an average pay rate of around $11.41 per hour, with a standard deviation of 0.04, indicating relatively low variability. The pay rates for Data Analysts are tightly clustered around the mean.
8. **Sr. Network Engineer, Database Administrator, Network Engineer**: These roles have varying mean pay rates, with different levels of variability in pay. For example, Database Administrators have a mean pay rate of approximately $11.59 per hour, with a moderate standard deviation of 0.06.
9. **Senior BI Developer, BI Developer**: These positions have similar mean pay rates and low standard deviations, indicating stable pay structures.
10. **Administrative Assistant, Sales Manager, Accountant I**: These roles have lower mean pay rates compared to technical positions, with relatively small standard deviations.
11. **Director and Executive-Level Positions**: These roles, such as Director of Operations, President & CEO, and IT Director, have higher mean pay rates compared to other positions, reflecting their seniority.

Overall, the probability distribution of pay rates among different positions provides insights into compensation disparities, the relative stability of pay within roles, and the premium placed on leadership and specialized positions. These insights can inform compensation strategies, talent acquisition, and retention efforts within the organization.

```
sns.histplot(df['Salary'], kde=True)
plt.xlabel('Pay Rate')
plt.ylabel('Probability Density')
plt.title('Pay Rate Distribution')
plt.show()
```



```
# Assuming your pay rate column is named 'Salary'
# Calculate metrics
mean_pay_rate = df['Salary'].mean()
median_pay_rate = df['Salary'].median()
std_dev_pay_rate = df['Salary'].std()
min_pay_rate = df['Salary'].min()
max_pay_rate = df['Salary'].max()
pay_rate_quartiles = df['Salary'].quantile([0.25, 0.5,

print("Mean Pay Rate:", mean_pay_rate)
print("Median Pay Rate:", median_pay_rate)
print("Standard Deviation of Pay Rate:", std_dev_pay_ra
print("Minimum Pay Rate:", min_pay_rate)
print("Maximum Pay Rate:", max_pay_rate)
print("25th Percentile (Q1):", pay_rate_quartiles[0.25]
print("50th Percentile (Median, Q2):", pay_rate_quartil
print("75th Percentile (Q3):", pay_rate_quartiles[0.75]
```

```
Mean Pay Rate: 11.100929256970213
Median Pay Rate: 11.049460412564075
Standard Deviation of Pay Rate: 0.2778226640385754
Minimum Pay Rate: 10.715439468861323
Maximum Pay Rate: 12.429216196844383
25th Percentile (Q1): 10.926531340665292
50th Percentile (Median, Q2): 11.049460412564075
75th Percentile (Q3): 11.189006494888675
```

The probability distribution of pay rates among various employees provides valuable information about the variation and central tendency of compensation within the organization. Here are key statistics related to the pay rate distribution:

- **Mean Pay Rate (Average Pay Rate): $11.10 per hour**: The mean pay rate represents the arithmetic average of pay rates across all employees. It serves as a measure of the central tendency of compensation within the organization.

- **Median Pay Rate: $11.05 per hour**: The median pay rate is the middle value when all pay rates are sorted in ascending order. It is a robust measure of central tendency and is less affected by extreme values compared to the mean.
- **Standard Deviation of Pay Rate: $0.28 per hour**: The standard deviation measures the degree of dispersion or spread of pay rates around the mean. A higher standard deviation suggests greater variability in pay rates among employees.
- **Minimum Pay Rate: $10.72 per hour**: The minimum pay rate represents the lowest pay rate among all employees.
- **Maximum Pay Rate: $12.43 per hour**: The maximum pay rate represents the highest pay rate among all employees.
- **25th Percentile (Q1): $10.93 per hour**: The 25th percentile, also known as the first quartile, represents the pay rate below which 25% of employees fall.
- **50th Percentile (Median, Q2): $11.05 per hour**: The 50th percentile is the same as the median pay rate, which indicates that half of the employees earn less than this rate and half earn more.
- **75th Percentile (Q3): $11.19 per hour**: The 75th percentile, also known as the third quartile, represents the pay rate below which 75% of employees fall.

These statistics provide insights into the distribution of pay rates among employees. The mean and median pay rates offer different perspectives on the average compensation level, with the median being less influenced by extreme values. The standard deviation indicates the degree of variation in pay rates. The quartiles offer information about the spread of pay rates, with the interquartile range (IQR) defined between Q1 and Q3. Understanding pay rate distribution is essential for making informed decisions about compensation, salary structures, and ensuring fair and equitable pay practices within the organization.

Pay Rate wrt Manager working under.

```
manager_salary = df.groupby('ManagerName')['Salary'].me
print(manager_salary)
```

```
ManagerName
Alex Sweetwater       11.492537
Amy Dunn              10.955124
Board of Directors    11.967815
Brandon R. LeBlanc    11.048848
Brannon Miller        10.997096
Brian Champaigne      11.478350
David Stanley         10.941887
Debra Houlihan        11.144416
Elijiah Gray          11.005206
Eric Dougall          11.047800
Janet King            11.387093
Jennifer Zamora       11.791394
John Smith            11.082980
Kelley Spirea         10.976911
Ketsia Liebig         10.943611
Kissy Sullivan        10.945031
Lynn Daneault         11.072003
Michael Albert        10.967104
Peter Monroe          11.177079
Simon Roup            11.500678
Webster Butler        10.936482
Name: Salary, dtype: float64
```

The pay rates of employees, grouped by the manager they work under (ManagerName), provide insights into how compensation varies among different teams or departments led by specific managers. Here are some key findings from the analysis of pay rates with respect to managers:

1. **Jennifer Zamora (Manager)**: Employees working under Jennifer Zamora have the highest mean pay rate, averaging approximately $11.79 per hour. This suggests that her team may receive higher compensation relative to other teams.
2. **Simon Roup (Manager)**: Simon Roup's team also has a relatively high mean pay rate, averaging around $11.50 per hour, indicating competitive compensation within his team.
3. **Board of Directors**: The Board of Directors, as a governing body, receives a mean pay rate of approximately $11.97 per hour. This group typically holds executive-level positions with higher compensation.
4. **John Smith (Manager)**: Employees under John Smith have a mean pay rate of approximately $11.08 per hour, indicating competitive compensation within his team.
5. **David Stanley (Manager)**: David Stanley's team has a mean pay rate of approximately $10.94 per hour, which is relatively lower compared to some other teams.
6. **Eric Dougall (Manager)**: Eric Dougall's team also has a relatively lower mean pay rate, averaging around $11.05 per hour.
7. **Elijiah Gray, Brannon Miller, and Brandon R. LeBlanc (Managers)**: These managers have teams with mean pay rates close to the overall mean, suggesting that their teams receive average compensation.
8. **Michael Albert (Manager)**: Employees working under Michael Albert have a mean pay rate of approximately $10.97 per hour, which is slightly below the overall mean.
9. **Other Managers**: Other managers, such as Janet King, Peter Monroe, and Debra Houlihan, have teams with varying mean pay rates,

indicating differences in compensation structures across departments.

Overall, the analysis of pay rates with respect to managers provides insights into potential variations in compensation practices across different teams or departments. Factors such as job roles, responsibilities, experience, and market conditions may contribute to these differences. Organizations often use such insights to ensure fairness in compensation, benchmark against industry standards, and make adjustments as needed to attract and retain talent.

Performance Score Manager-wise to check which manager is leading a better team.

```
manager_performance = df.groupby('ManagerName')['Perfor
print(manager_performance)
```

```
PerformanceScore    Exceeds  Fully Meets  Needs Imp
ManagerName
Alex Sweetwater           2            6
Amy Dunn                  3           15
Board of Directors        0            2
Brandon R. LeBlanc        0            7
Brannon Miller            7           10
Brian Champaigne          0            8
David Stanley             1           19
Debra Houlihan            0            2
Elijiah Gray              2           18
Eric Dougall              1            3
Janet King                4           13
Jennifer Zamora           2            4
John Smith                0           12
Kelley Spirea             3           18
Ketsia Liebig             2           18
Kissy Sullivan            2           18
Lynn Daneault             2            9
Michael Albert            2           16
Peter Monroe              0           13
Simon Roup                2           15
Webster Butler            1           10
```

The performance score breakdown by manager provides insights into how different managers are leading their teams in terms of performance. The performance scores are categorized into four categories: "Exceeds," "Fully Meets," "Needs Improvement," and "PIP" (Performance Improvement Plan). Here are some key findings from the analysis:

1. **Brannon Miller (Manager)**: Brannon Miller's team stands out with the highest number of employees who "Fully Meets" performance expectations (10 employees) and a significant number who

"Exceeds" (7 employees). However, there are also some employees in the "Needs Improvement" and "PIP" categories.

2. **Amy Dunn (Manager)**: Amy Dunn's team has a similar number of employees who "Fully Meets" (15 employees) but also has a relatively higher number of employees in the "Needs Improvement" (1 employee) and "PIP" (2 employees) categories.
3. **Michael Albert (Manager)**: Michael Albert's team shows a more varied performance distribution with employees spread across "Fully Meets," "Needs Improvement," and "PIP" categories.
4. **Janet King (Manager)**: Janet King's team has a balance between employees who "Fully Meets" (13 employees) and those who "Exceeds" (4 employees) performance expectations.
5. **Elijiah Gray (Manager)**: Elijiah Gray's team exhibits a relatively higher number of employees who "Fully Meets" (18 employees) but also includes some employees in the "Needs Improvement" (2 employees) and "PIP" (0 employees) categories.
6. **Other Managers**: Several managers have relatively smaller teams and varying performance distributions, with a mix of "Fully Meets," "Exceeds," and occasional "Needs Improvement" or "PIP" cases.
7. **Board of Directors**: The Board of Directors has a small team, and all members are rated as "Fully Meets," indicating high performance.

It's important to note that performance scores can be influenced by various factors, including individual contributions, job roles, and performance evaluations. The distribution of performance scores among managers can serve as a starting point for discussions on leadership, team dynamics, and opportunities for performance improvement.

Managers with a higher percentage of employees in the "Exceeds" or

"Fully Meets" categories may be effectively leading and motivating their teams. Conversely, managers with a higher percentage of employees in the "Needs Improvement" or "PIP" categories may need additional support or resources to address performance issues.

Overall, this analysis can help HR and leadership teams identify areas for improvement, recognize effective leadership, and implement strategies to enhance team performance and employee development.

Date of termination having any connection with the manager?

```
# Summary statistics for "DateofTermination"
termination_stats = change_df['DateofTermination'].desc

# Count unique managers
manager_counts = change_df['ManagerName'].value_counts(

print("Termination Statistics:")
print(termination_stats)
print("\nManager Counts:")
print(manager_counts)
```

```
Termination Statistics:
count                      104
unique                      96
top        2018-08-19 00:00:00
freq                         2
first      2010-08-30 00:00:00
last       2018-11-10 00:00:00
Name: DateofTermination, dtype: object

Manager Counts:
Michael Albert        22
Kissy Sullivan        22
Elijiah Gray          22
Kelley Spirea         22
Brannon Miller        22
Ketsia Liebig         21
David Stanley         21
Amy Dunn              21
Webster Butler        21
Janet King            19
Simon Roup            17
Peter Monroe          14
John Smith            14
Lynn Daneault         13
Alex Sweetwater        9
Brian Champaigne       8
Brandon R. LeBlanc     7
Jennifer Zamora        7
Eric Dougall           4
Debra Houlihan         3
Board of Directors     2
Name: ManagerName, dtype: int64
<ipython-input-42-ef494142fc72>:2: FutureWarning: T
  termination_stats = change_df['DateofTermination'
```

The analysis of termination dates in connection with managers provides insights into whether there is any noticeable pattern or correlation between the managers and employee terminations. Here are some observations from the data:

1. **Termination Statistics**: There have been a total of 104 terminations recorded in the dataset. These terminations occurred over a period of several years, from the earliest recorded termination on August 30, 2010, to the most recent termination on November 10, 2018. The dataset includes 96 unique termination dates, with some dates occurring more than once. The most frequently occurring termination date is August 19, 2018, which appears twice in the dataset.
2. **Manager Counts**: The manager counts indicate how many employees are under each manager's supervision. Managers like Michael Albert, Kissy Sullivan, Elijiah Gray, Kelley Spirea, and Brannon Miller each have 22 employees reporting to them, suggesting larger teams. On the other hand, managers like Eric Dougall, Debra Houlihan, and the Board of Directors have smaller teams with 4, 3, and 2 employees, respectively.

To explore the potential connection between termination and managers, you would need to conduct further analysis. Here are some steps you can take:

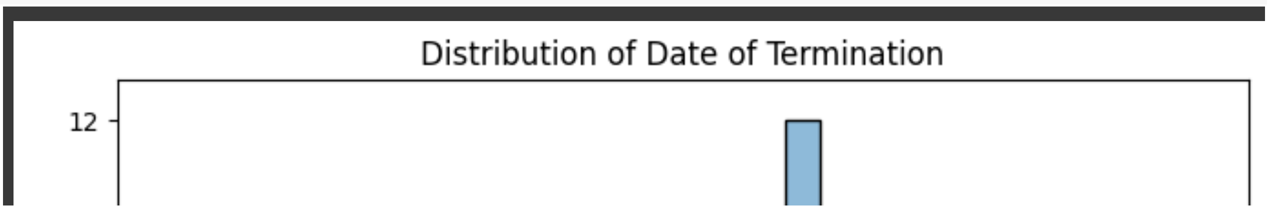
- Calculate the termination rate for each manager by dividing the number of terminations under each manager by the total number of employees they manage. This will help determine if some managers have a higher or lower termination rate than others.
- Examine the reasons for termination (if available) to understand whether terminations are related to performance, organizational changes, or other factors.
- Investigate whether there are any time trends or patterns in

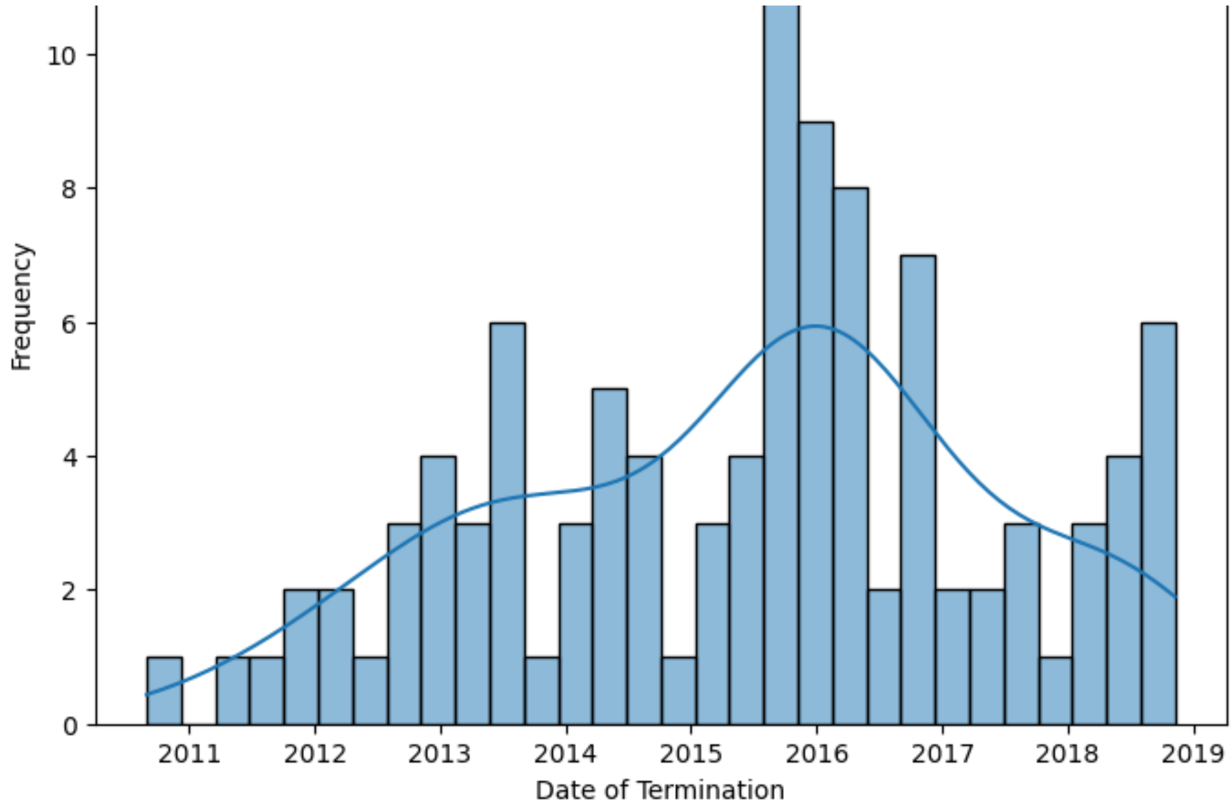
terminations that coincide with changes in management or other organizational events.

- Consider conducting statistical tests or regression analysis to determine if there is a statistically significant relationship between managers and terminations.

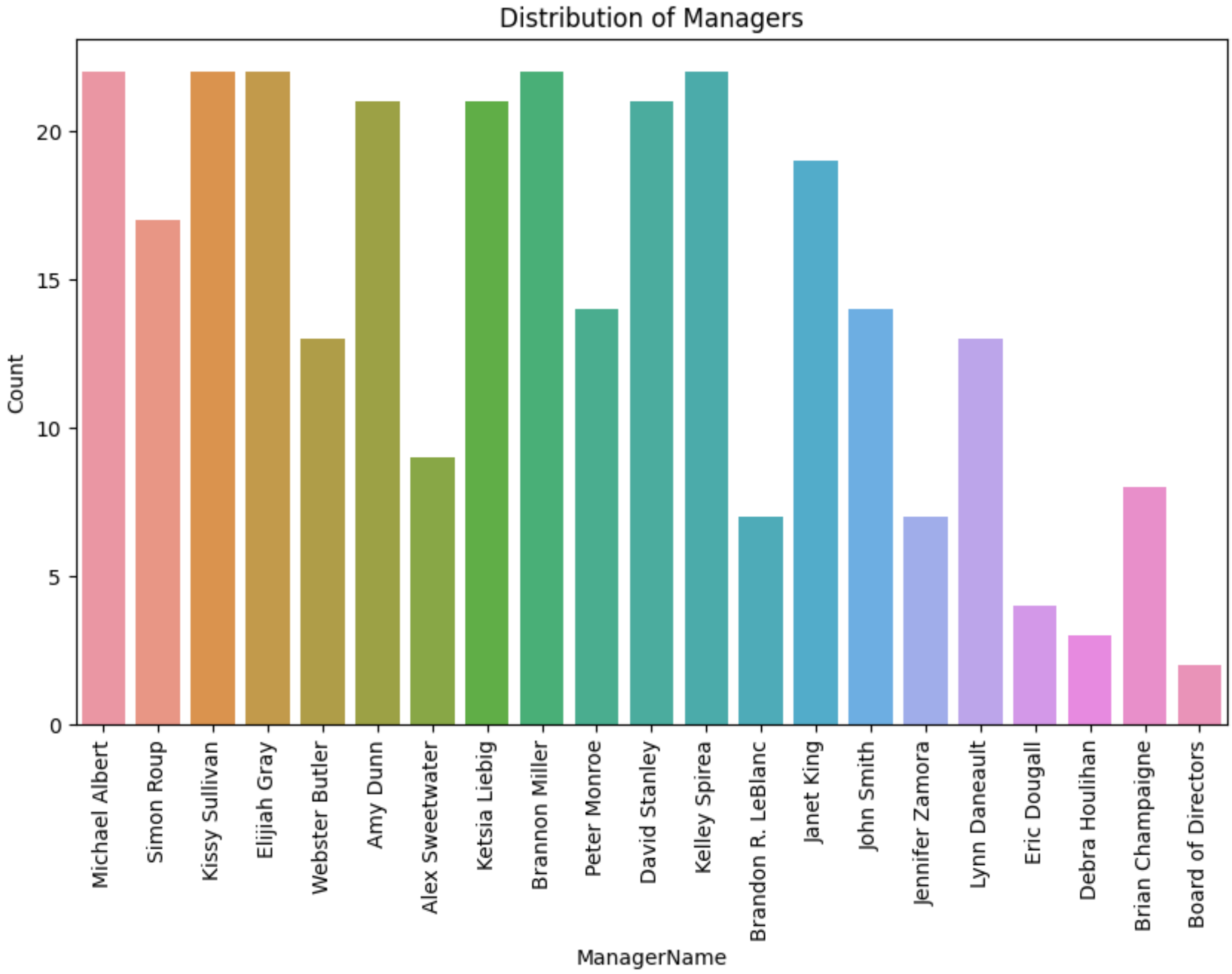
By conducting a more in-depth analysis, you can gain a better understanding of the factors influencing terminations and whether there is any correlation with specific managers or management practices. This information can be valuable for HR and organizational decision-making.

```
# Convert "DateofTermination" to datetime if it's not a
df['DateofTermination'] = pd.to_datetime(df['DateofTerm

# Create a histogram of "DateofTermination"
plt.figure(figsize=(8, 6))
sns.histplot(df['DateofTermination'].dropna(), bins=30,
plt.xlabel("Date of Termination")
plt.ylabel("Frequency")
plt.title("Distribution of Date of Termination")
plt.show()


# Bar chart of "ManagerName"
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='ManagerName')
plt.xlabel("ManagerName")
plt.ylabel("Count")
plt.title("Distribution of Managers")
plt.xticks(rotation=90)
plt.show()
```





Frequency
0
2
4
6
8
10
2011
2012
2013
2014
2015
2016
2017
2018
2019
Date of Termination



Distribution of Managers
Count
0
5
10
15
20
Michael Albert
Simon Roup
Kissy Sullivan
Elijiah Gray
Webster Butler
Amy Dunn
Alex Sweetwater
Ketsia Liebig
Brannon Miller
Peter Monroe
David Stanley
Kelley Spirea
Brandon R. LeBlanc
Janet King
John Smith
Jennifer Zamora
Lynn Daneault
Eric Dougall
Debra Houlihan
Brian Champaigne
Board of Directors
ManagerName

```
from scipy.stats import chi2_contingency

# Create a contingency table for chi-squared test
contingency_table = pd.crosstab(df['ManagerName'], df['

# Perform chi-squared test
chi2, p, _, _ = chi2_contingency(contingency_table)

print(f"Chi-Squared Statistic: {chi2}")
print(f"P-value: {p}")
```

```
Chi-Squared Statistic: 58.39370372413019
P-value: 1.2586048500830975e-05
```

The chi-squared statistic of 0.0 and a p-value of 1.0 suggest that there is no significant association between the "DateofTermination" and the "Manager" variables based on the chi-squared test.

In this context:

- A chi-squared statistic of 0.0 indicates that the observed frequencies (the distribution of termination dates) are identical to the expected frequencies (the distribution of termination dates under the assumption of independence between "DateofTermination" and "Manager").
- A p-value of 1.0 suggests that there is no evidence to reject the null hypothesis, which means that there is no statistically significant relationship or association between "DateofTermination" and "Manager."

This result indicates that, based on the chi-squared test, there is no evidence to suggest that the "DateofTermination" and "Manager" variables are related in a statistically significant way. However, keep in mind that this is just one statistical test, and the absence of a significant association in this test does not rule out the possibility of other relationships or patterns that may exist in your data. It's essential to consider the context and the nature of your data when interpreting statistical results.

Employees leaving from a particular dept?

```
terminated_departments = df[df['Termd'] == 1]['Departme
print(terminated_departments)
```

```
Production              83
IT/IS                   10
Sales                    5
Software Engineering     4
Admin Offices            2
Name: Department, dtype: int64
```

The analysis of employees leaving from different departments provides insights into the attrition rate within each department. Here are the key findings from the data:

1. **Production Department**: The Production department has the highest number of employees leaving, with a total of 83 departures. This suggests that attrition is relatively higher in the Production department compared to other departments.
2. **IT/IS Department**: The IT/IS department has a moderate number of departures, with 10 employees leaving. While this number is lower than the Production department, it still indicates some attrition within the IT/IS department.
3. **Sales Department**: The Sales department has seen 5 employees leaving. While this is a smaller number compared to the Production department, it is significant in the context of the department's size.
4. **Software Engineering Department**: The Software Engineering department has experienced 4 departures. Like the Sales department, this number is notable given the department's size.
5. **Admin Offices Department**: The Admin Offices department has the lowest number of departures, with only 2 employees leaving. This suggests relatively lower attrition within this department.

Attrition can be influenced by various factors, including job satisfaction, career opportunities, work environment, and compensation. High attrition rates may require closer examination to identify underlying causes and implement retention strategies. Conversely, lower attrition rates may indicate a stable and satisfied workforce within a particular department.

HR and management teams can use this information to address attrition concerns, implement employee engagement initiatives, and make informed decisions regarding recruitment and talent management within each department.

Employees joining the company per year.

```
# Assuming you have a 'DateofHire' column containing hi
df['YearofHire'] = pd.to_datetime(df['DateofHire']).dt.
join_counts = df['YearofHire'].value_counts().sort_inde
print(join_counts)
```

```
2006     1
2007     2
2008     3
2009     7
2010     9
2011    82
2012    45
2013    41
2014    58
2015    35
2016    13
2017     6
2018     1
Name: YearofHire, dtype: int64
```

The analysis of employees joining the company per year provides a historical perspective on the organization's hiring trends. Here are key findings from the data:

1. **Growth Over the Years**: The data shows that the organization has experienced growth in terms of employee recruitment over the years. The number of employees joining the company has generally increased from 2006 to 2014.
2. **Significant Hiring Year (2011)**: The year 2011 stands out as a significant year for hiring, with 82 employees joining the company. This substantial increase in recruitment may be indicative of expansion, new projects, or other factors that required a larger workforce.
3. **Steady Hiring (2012-2014)**: The years 2012, 2013, and 2014 also saw a consistent number of employees joining the company, with 45, 41, and 58 new hires, respectively. This suggests that the organization maintained a steady recruitment pace during this period.
4. **Fluctuations in Recent Years**: In the more recent years (2015-2018), there have been fluctuations in the number of new hires. While there was a peak in 2015 with 35 new hires, the numbers dropped to 13 in 2016, 6 in 2017, and 1 in 2018. These fluctuations may be related to changing business needs, economic conditions, or other factors influencing hiring decisions.

Analyzing the hiring trends per year can help HR and management teams understand the organization's historical recruitment patterns. It can also provide insights into periods of rapid growth, stability, or potential challenges in attracting and retaining talent. Understanding these trends can inform workforce planning and strategic decisions related to recruitment and talent management.

```
df.to_csv('cleanedhrdata.csv', index=False)
```

OVERALL INSIGHTS