Data Viz in Plotting

In [ ]:
```
import pandas as pd  # For data manipulation and analysis
import numpy as np  # For numerical computations
import matplotlib.pyplot as plt  # For creating static, animated, and interactive visualizations
import seaborn as sns  # For statistical data visualization based on Matplotlib
import scipy  # For scientific and technical computing (including optimization, integration, and statistics)
```

In [4]:
```
import pandas as pd
import numpy as np
# Creating realistic data for employees
data = {
    'Employee ID': np.arange(1001, 1011),
    'Employee Name': ['Satender Kumar', 'data 1', 'Jane Smith', 'Robert Brown', 'Emily Davis', 'Michael Wilson'
    'Department': ['Data Analyst', 'IT', 'Finance', 'Marketing', 'Sales', 'Operations', 'R&D', 'Support', 'Admi
    'Age': [24, np.random.randint(25, 60), np.random.randint(25, 60), np.random.randint(25, 60), np.random.rand
    'Location': ['London, Canada', 'Toronto', 'London', 'Sydney', 'San Francisco', 'Paris', 'Berlin', 'Tokyo',
    'Salary': np.random.randint(50000, 150000, size=10),
    'Years with Company': np.random.randint(1, 15, size=10),
    'Position': ['Data Analyst', 'Developer', 'Analyst', 'Designer', 'Consultant', 'Engineer', 'Scientist', 'Su
    'Performance Score': np.random.randint(1, 5, size=10),
    'Bonus': np.random.randint(1000, 10000, size=10),
    'Gender': ['Male', 'Male', 'Female', 'Male', 'Female', 'Male', 'Female', 'Male', 'Female', 'Male'],
    'Marital Status': ['Single', 'Single', 'Married', 'Single', 'Single', 'Married', 'Married', 'Single', 'Marr
    'Education': ['Bachelor', 'Master', 'PhD', 'Bachelor', 'Master', 'PhD', 'Bachelor', 'Master', 'Bachelor', '
    'Hire Date': pd.to_datetime(['2019-06-12', '2015-07-23', '2012-09-05', '2018-11-30', '2013-05-19', '2019-02
    'Overtime Hours': np.random.randint(0, 20, size=10),
    'Sick Days Taken': np.random.randint(0, 10, size=10),
    'Vacation Days Taken': np.random.randint(5, 20, size=10),
    'Training Hours': np.random.randint(10, 50, size=10),
    'Certifications': ['Yes', 'No', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'No', 'Yes'],
    'Supervisor': ['Anna Smith', 'Brian Adams', 'Clara Jones', 'Daniel Martin', 'Eva Rodriguez', 'Frank Bell',
}

# Creating the DataFrame
df = pd.DataFrame(data)
```

In [10]:
```
df
```

Out[10]:

	Employee ID	Employee Name	Department	Age	Location	Salary	Years with Company	Position	Performance Score	Bonus	Gender	Marital Status	Educat
0	1001	Satender Kumar	Data Analyst	24	London, Canada	50379	3	Data Analyst	2	6598	Male	Single	Bach
1	1002	data 1	IT	37	Toronto	98030	3	Developer	4	6601	Male	Single	Ma
2	1003	Jane Smith	Finance	42	London	54742	13	Analyst	3	1951	Female	Married	F
3	1004	Robert Brown	Marketing	50	Sydney	79468	7	Designer	4	3987	Male	Single	Bach
4	1005	Emily Davis	Sales	39	San Francisco	140576	5	Consultant	1	6045	Female	Single	Ma
5	1006	Michael Wilson	Operations	55	Paris	149810	2	Engineer	2	2484	Male	Married	F
6	1007	Sarah Taylor	R&D	46	Berlin	122118	1	Scientist	2	1421	Female	Married	Bach
7	1008	David Lee	Support	56	Tokyo	121107	11	Support Agent	4	3473	Male	Single	Ma
8	1009	Laura Johnson	Admin	38	Dubai	143323	8	Admin Assistant	1	5515	Female	Married	Bach
9	1010	James White	Legal	48	Singapore	51334	12	Lawyer	1	3612	Male	Single	Ma

In [11]:
```
import pandas as pd
import numpy as np
```

In [13]:
```
# Creating realistic data for a second set of employees
data1 = {
    'Employee ID': np.arange(1011, 1021),
    'Employee Name': ['Satender Kumar', 'data 1', 'Chris Evans', 'Natalie Portman', 'Tom Holland', 'Emma Watson
    'Department': ['Data Analyst', 'HR', 'IT', 'Marketing', 'Finance', 'Sales', 'R&D', 'Operations', 'Legal', '
    'Age': [24, np.random.randint(25, 60), np.random.randint(25, 60), np.random.randint(25, 60), np.random.rand
```

```
    'Location': ['London, Canada', 'Los Angeles', 'New York', 'Chicago', 'Houston', 'Phoenix', 'Philadelphia',
    'Salary': np.random.randint(60000, 160000, size=10),
    'Years with Company': np.random.randint(1, 20, size=10),
    'Position': ['Data Analyst', 'HR Manager', 'IT Specialist', 'Marketing Coordinator', 'Financial Analyst', '
    'Performance Score': np.random.randint(1, 5, size=10),
    'Bonus': np.random.randint(2000, 12000, size=10),
    'Gender': ['Male', 'Male', 'Female', 'Female', 'Male', 'Female', 'Male', 'Female', 'Male', 'Male'],
    'Marital Status': ['Single', 'Married', 'Single', 'Single', 'Married', 'Single', 'Single', 'Married', 'Sing
    'Education': ['Master', 'Bachelor', 'Master', 'PhD', 'Bachelor', 'Master', 'PhD', 'Bachelor', 'Master', 'Phl
    'Hire Date': pd.to_datetime(['2018-07-15', '2014-03-22', '2011-10-12', '2017-04-17', '2015-09-23', '2016-11
    'Overtime Hours': np.random.randint(0, 25, size=10),
    'Sick Days Taken': np.random.randint(0, 8, size=10),
    'Vacation Days Taken': np.random.randint(7, 22, size=10),
    'Training Hours': np.random.randint(15, 55, size=10),
    'Certifications': ['Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'No'],
    'Supervisor': ['John Smith', 'Michael Johnson', 'Patricia Williams', 'Linda Brown', 'Barbara Jones', 'Elizal
}

# Creating the second DataFrame
df1 = pd.DataFrame(data1)
```

In [15]:
```
df1
```

Out[15]:

	Employee ID	Employee Name	Department	Age	Location	Salary	Years with Company	Position	Performance Score	Bonus	Gender	Marital Status	Edu
0	1011	Satender Kumar	Data Analyst	24	London, Canada	95392	5	Data Analyst	2	3416	Male	Single	
1	1012	data 1	HR	56	Los Angeles	75471	17	HR Manager	1	9690	Male	Married	B
2	1013	Chris Evans	IT	30	New York	129083	12	IT Specialist	2	4132	Female	Single	
3	1014	Natalie Portman	Marketing	27	Chicago	104039	3	Marketing Coordinator	3	10256	Female	Single	
4	1015	Tom Holland	Finance	56	Houston	138405	5	Financial Analyst	3	10453	Male	Married	B
5	1016	Emma Watson	Sales	35	Phoenix	145713	19	Sales Manager	3	11268	Female	Single	
6	1017	Daniel Radcliffe	R&D	41	Philadelphia	62592	1	Research Scientist	2	9166	Male	Single	
7	1018	Scarlett Johansson	Operations	44	San Antonio	119897	10	Operations Manager	3	8533	Female	Married	B
8	1019	Robert Downey Jr.	Legal	27	San Diego	92816	3	Legal Advisor	4	9062	Male	Single	
9	1020	Mark Ruffalo	Support	46	Dallas	144765	1	Support Specialist	3	4142	Male	Married	

In [17]:
```
merged_df = pd.merge(df, df1, on='Employee ID', suffixes=('_df', '_df1'), how='outer')
print(merged_df)
```

```
   Employee ID Employee Name_df Department_df  Age_df     Location_df  \
0         1001   Satender Kumar  Data Analyst    24.0  London, Canada
1         1002           data 1            IT    37.0         Toronto
2         1003       Jane Smith       Finance    42.0          London
3         1004     Robert Brown     Marketing    50.0          Sydney
4         1005      Emily Davis         Sales    39.0   San Francisco
5         1006   Michael Wilson    Operations    55.0           Paris
6         1007     Sarah Taylor           R&D    46.0          Berlin
7         1008        David Lee       Support    56.0           Tokyo
8         1009    Laura Johnson         Admin    38.0           Dubai
9         1010      James White         Legal    48.0       Singapore
10        1011              NaN           NaN     NaN             NaN
11        1012              NaN           NaN     NaN             NaN
12        1013              NaN           NaN     NaN             NaN
13        1014              NaN           NaN     NaN             NaN
14        1015              NaN           NaN     NaN             NaN
15        1016              NaN           NaN     NaN             NaN
16        1017              NaN           NaN     NaN             NaN
17        1018              NaN           NaN     NaN             NaN
18        1019              NaN           NaN     NaN             NaN
19        1020              NaN           NaN     NaN             NaN

   Salary_df  Years with Company_df      Position_df  Performance Score_df  \
0    50379.0                    3.0     Data Analyst                   2.0
1    98030.0                    3.0        Developer                   4.0
2    54742.0                   13.0          Analyst                   3.0
```

```
3     79468.0                   7.0         Designer                   4.0
4    140576.0                   5.0       Consultant                   1.0
5    149810.0                   2.0         Engineer                   2.0
6    122118.0                   1.0        Scientist                   2.0
7    121107.0                  11.0    Support Agent                   4.0
8    143323.0                   8.0  Admin Assistant                   1.0
9     51334.0                  12.0           Lawyer                   1.0
10        NaN                   NaN              NaN                   NaN
11        NaN                   NaN              NaN                   NaN
12        NaN                   NaN              NaN                   NaN
13        NaN                   NaN              NaN                   NaN
14        NaN                   NaN              NaN                   NaN
15        NaN                   NaN              NaN                   NaN
16        NaN                   NaN              NaN                   NaN
17        NaN                   NaN              NaN                   NaN
18        NaN                   NaN              NaN                   NaN
19        NaN                   NaN              NaN                   NaN

    Bonus_df  ... Gender_df1 Marital Status_df1 Education_df1 Hire Date_df1  \
0     6598.0  ...        NaN                NaN           NaN           NaT
1     6601.0  ...        NaN                NaN           NaN           NaT
2     1951.0  ...        NaN                NaN           NaN           NaT
3     3987.0  ...        NaN                NaN           NaN           NaT
4     6045.0  ...        NaN                NaN           NaN           NaT
5     2484.0  ...        NaN                NaN           NaN           NaT
6     1421.0  ...        NaN                NaN           NaN           NaT
7     3473.0  ...        NaN                NaN           NaN           NaT
8     5515.0  ...        NaN                NaN           NaN           NaT
9     3612.0  ...        NaN                NaN           NaN           NaT
10       NaN  ...       Male             Single        Master    2018-07-15
11       NaN  ...       Male            Married      Bachelor    2014-03-22
12       NaN  ...     Female             Single        Master    2011-10-12
13       NaN  ...     Female             Single           PhD    2017-04-17
14       NaN  ...       Male            Married      Bachelor    2015-09-23
15       NaN  ...     Female             Single        Master    2016-11-01
16       NaN  ...       Male             Single           PhD    2019-05-11
17       NaN  ...     Female            Married      Bachelor    2020-07-08
18       NaN  ...       Male             Single        Master    2013-08-19
19       NaN  ...       Male            Married           PhD    2012-01-09

    Overtime Hours_df1  Sick Days Taken_df1  Vacation Days Taken_df1  \
0                  NaN                  NaN                      NaN
1                  NaN                  NaN                      NaN
2                  NaN                  NaN                      NaN
3                  NaN                  NaN                      NaN
4                  NaN                  NaN                      NaN
5                  NaN                  NaN                      NaN
6                  NaN                  NaN                      NaN
7                  NaN                  NaN                      NaN
8                  NaN                  NaN                      NaN
9                  NaN                  NaN                      NaN
10                 3.0                  0.0                     18.0
11                18.0                  2.0                     14.0
12                16.0                  7.0                     16.0
13                 9.0                  6.0                     11.0
14                16.0                  1.0                     17.0
15                 2.0                  7.0                      9.0
16                17.0                  2.0                     21.0
17                21.0                  2.0                     13.0
18                 9.0                  3.0                     10.0
19                16.0                  7.0                     21.0

    Training Hours_df1 Certifications_df1     Supervisor_df1
0                  NaN                NaN                NaN
1                  NaN                NaN                NaN
2                  NaN                NaN                NaN
3                  NaN                NaN                NaN
4                  NaN                NaN                NaN
5                  NaN                NaN                NaN
6                  NaN                NaN                NaN
7                  NaN                NaN                NaN
8                  NaN                NaN                NaN
9                  NaN                NaN                NaN
10                42.0                Yes         John Smith
11                41.0                Yes    Michael Johnson
12                39.0                 No  Patricia Williams
13                36.0                Yes        Linda Brown
14                45.0                 No      Barbara Jones
15                51.0                Yes   Elizabeth Garcia
16                24.0                 No     Susan Martinez
17                37.0                Yes  Jessica Hernandez
18                16.0                Yes        Sarah Lopez
19                54.0                 No       Karen Wilson
```

[20 rows x 39 columns]

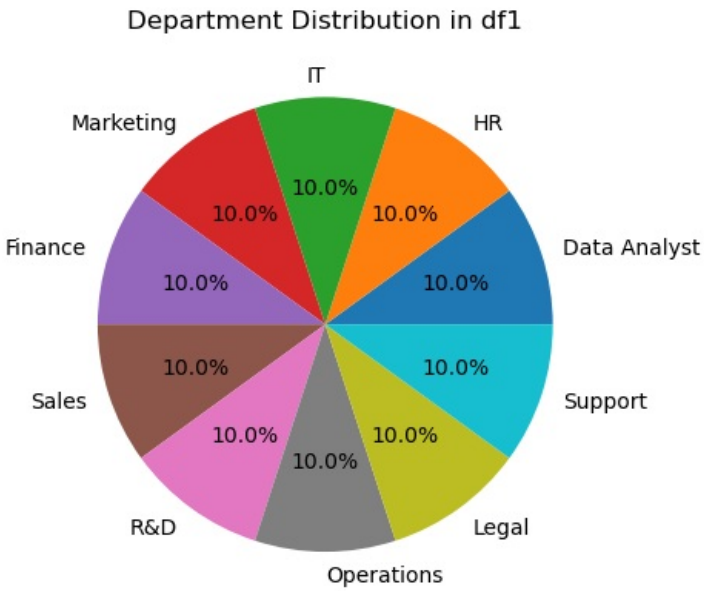
In [19]:
```
df.plot()
```

Out[19]: <Axes: >

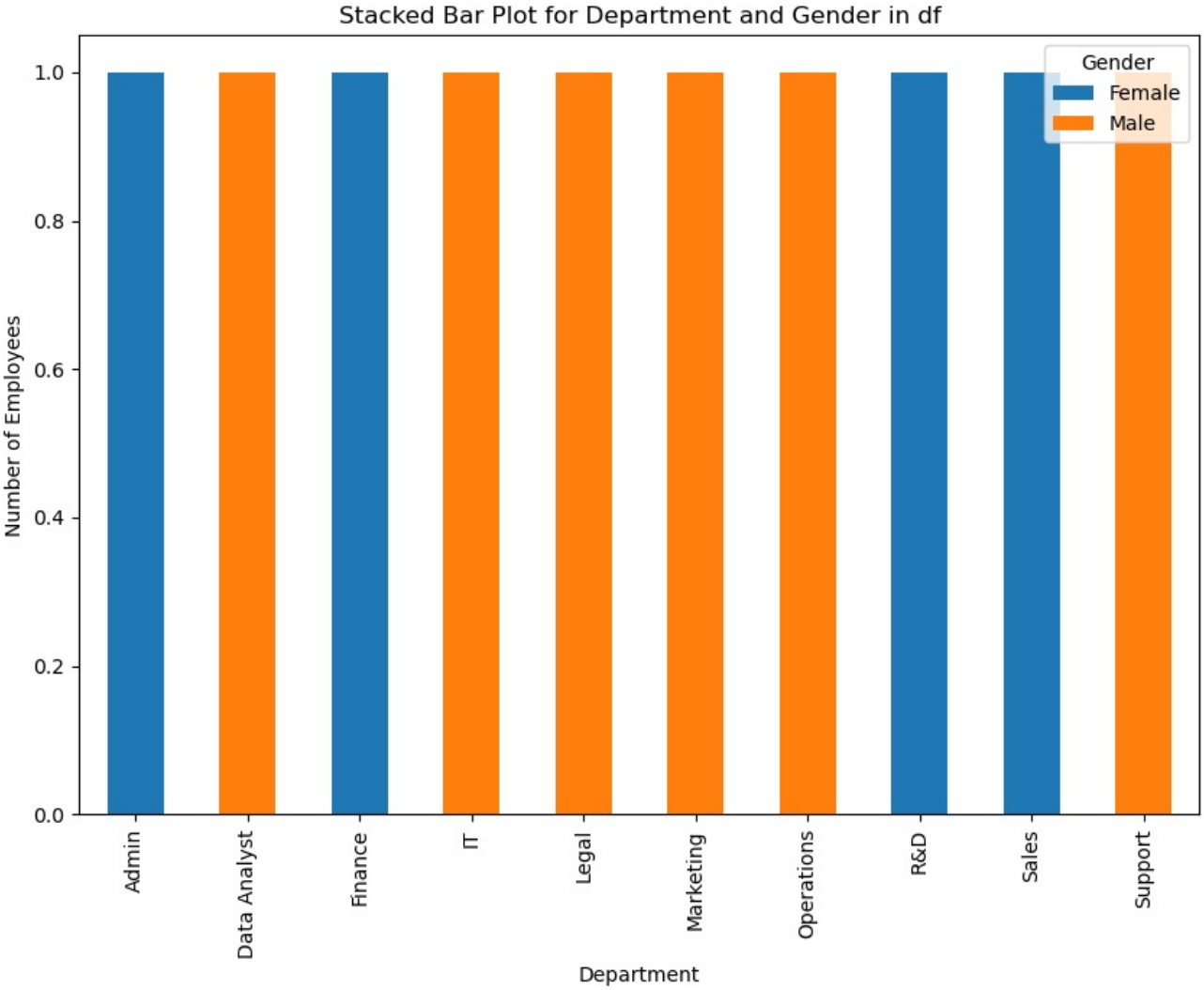
In [21]:
```
df1.plot()
```

Out[21]: <Axes: >

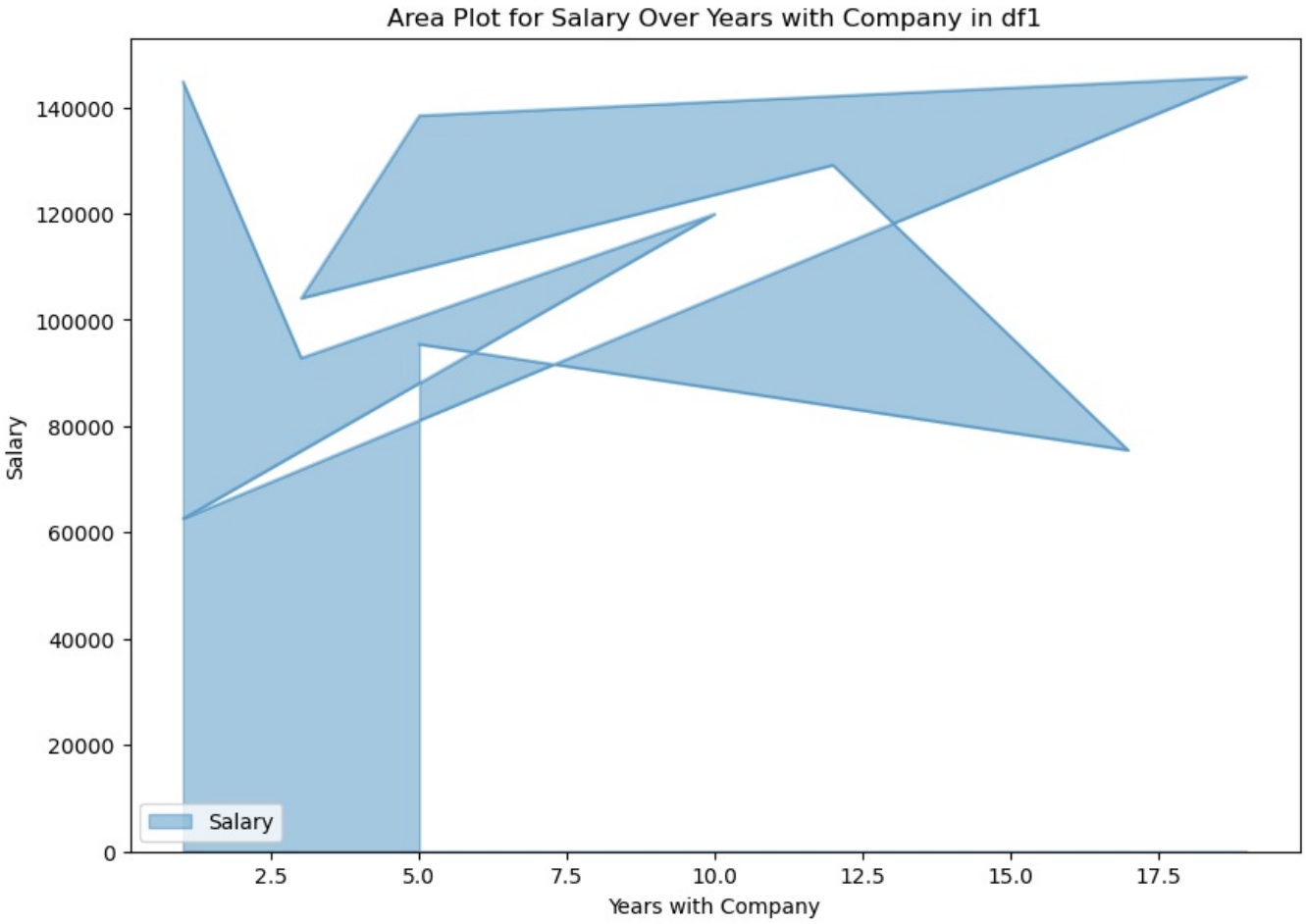
In [54]:
```
# Pie chart for department distribution in df1
df1['Department'].value_counts().plot(kind='pie', autopct='%1.1f%%', title='Department Distribution in df1')
plt.ylabel('')  # Remove the y-label for a cleaner look
plt.show()
```



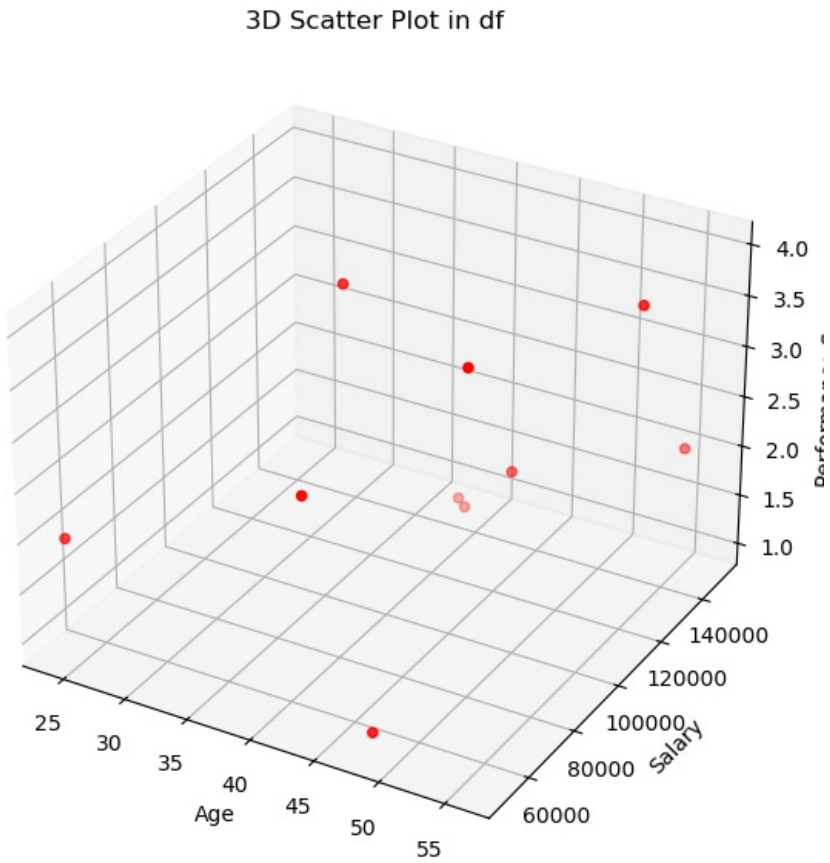
In [56]:
```
# Stacked bar plot showing count of employees by Department and Gender in df
df.groupby(['Department', 'Gender']).size().unstack().plot(kind='bar', stacked=True, figsize=(10, 7))
plt.title('Stacked Bar Plot for Department and Gender in df')
plt.ylabel('Number of Employees')
plt.show()
```



In [57]:
```
# Area plot showing salary over years with company in df1
df1.plot.area(x='Years with Company', y='Salary', figsize=(10, 7), alpha=0.4)
plt.title('Area Plot for Salary Over Years with Company in df1')
plt.ylabel('Salary')
plt.show()
```

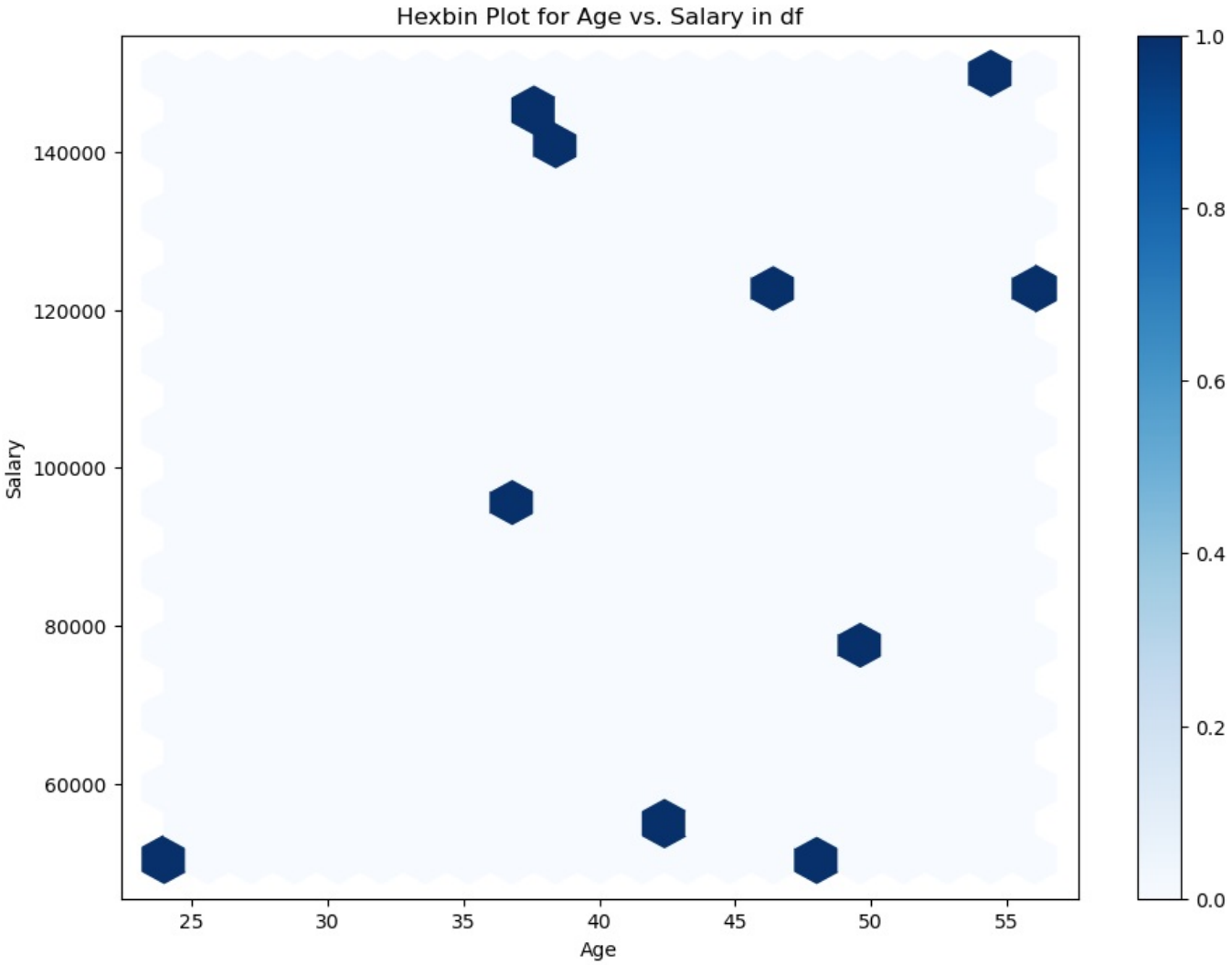


In [58]:
```
from mpl_toolkits.mplot3d import Axes3D

# 3D scatter plot in df
fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df['Age'], df['Salary'], df['Performance Score'], c='r', marker='o')
ax.set_xlabel('Age')
ax.set_ylabel('Salary')
ax.set_zlabel('Performance Score')
plt.title('3D Scatter Plot in df')
plt.show()
```



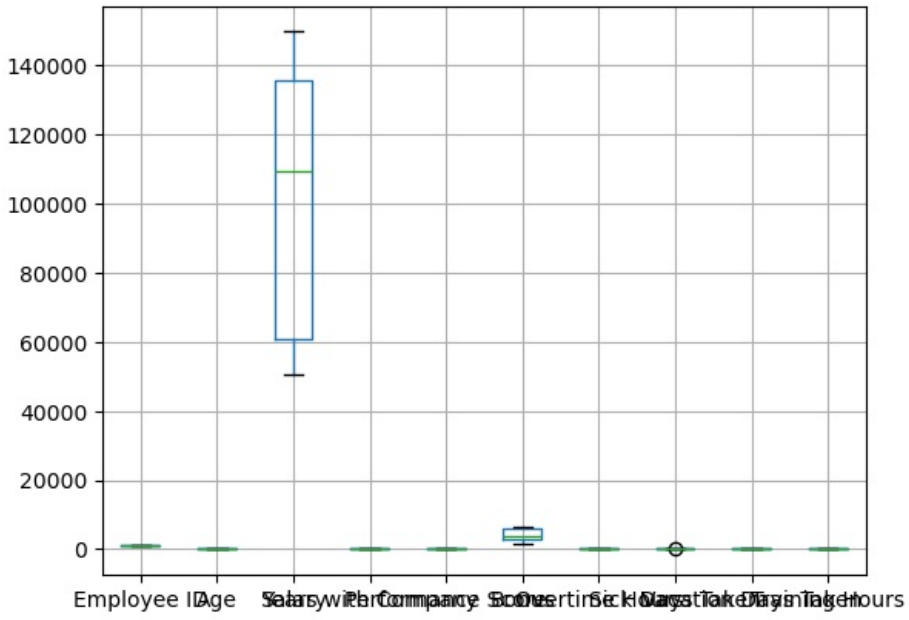
In [59]:
```
# Hexbin plot for Age vs. Salary in df1
df1.plot.hexbin(x='Age', y='Salary', gridsize=20, cmap='Blues', figsize=(10, 7))
plt.title('Hexbin Plot for Age vs. Salary in df1')
plt.show()
```

In [61]:
```
# Hexbin plot for Age vs. Salary in df
df.plot.hexbin(x='Age', y='Salary', gridsize=20, cmap='Blues', figsize=(11, 8))
plt.title('Hexbin Plot for Age vs. Salary in df')
plt.show()
```



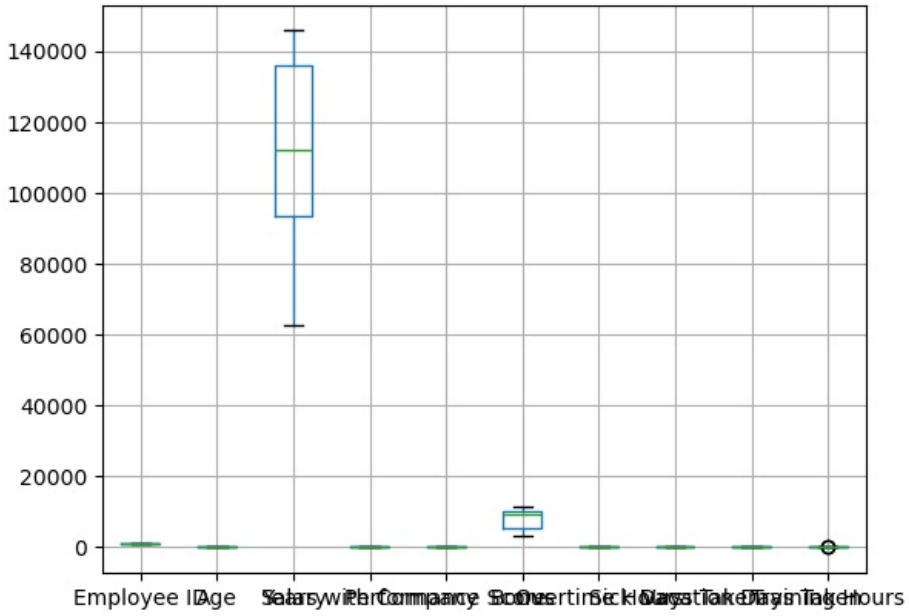
In [63]:
```
df.boxplot()
```

Out[63]: <Axes: >



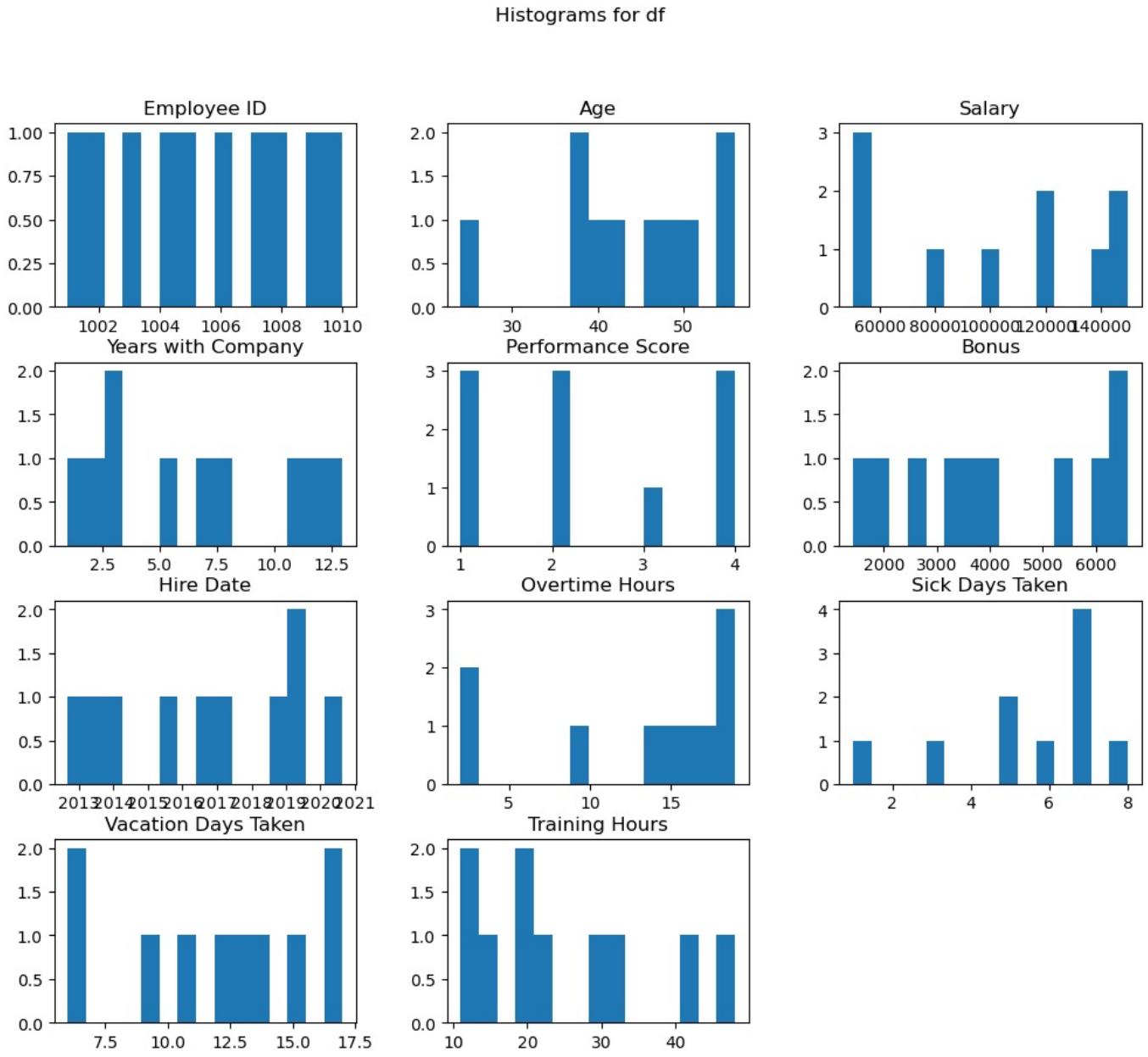
In [30]:
```
df1.boxplot()
```

Out[30]: <Axes: >

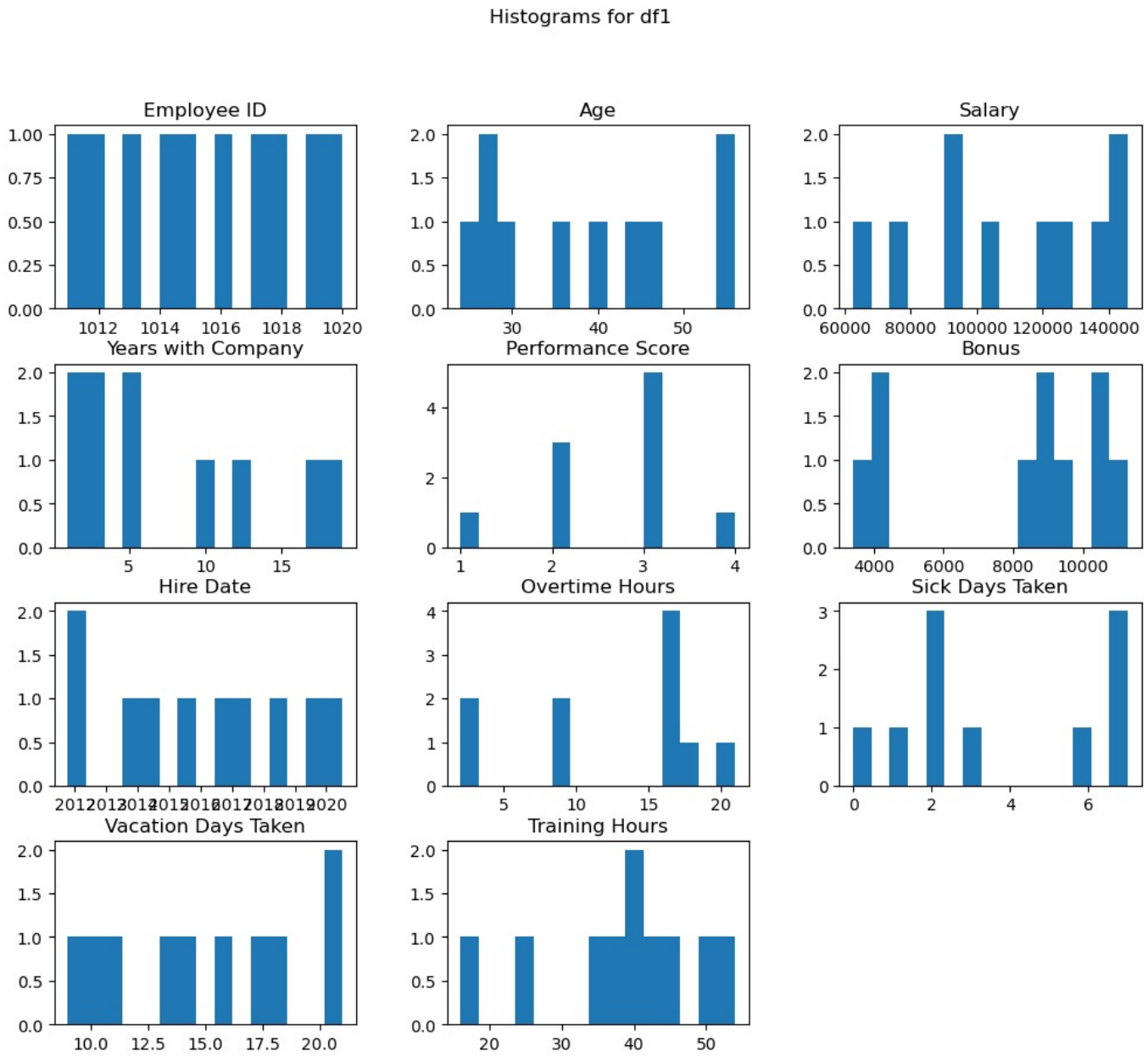


In [32]:

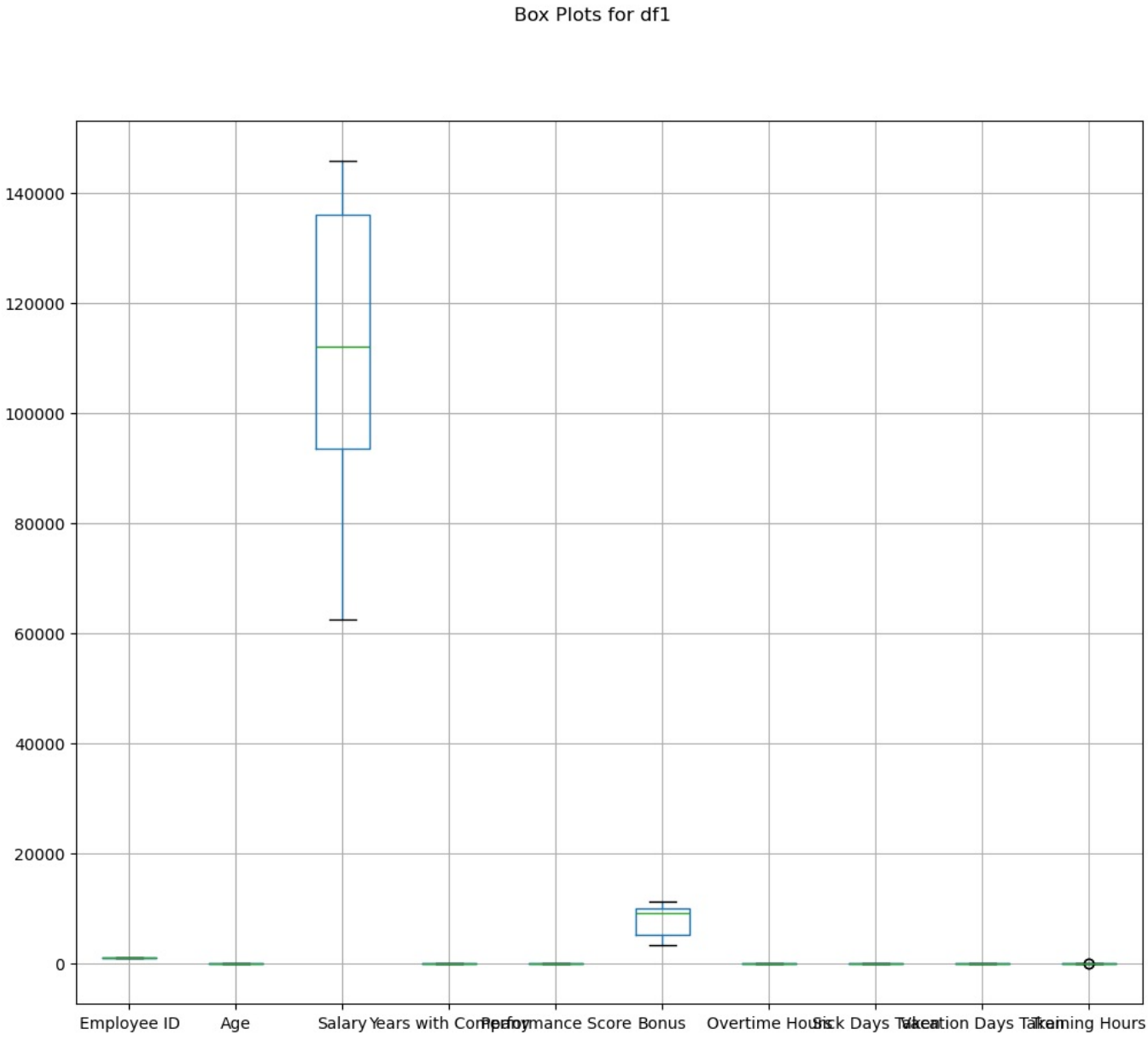
```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming df is your first DataFrame
df.hist(figsize=(12, 10), bins=15, grid=False)

# Display the plots for df
plt.suptitle('Histograms for df')
plt.show()
```



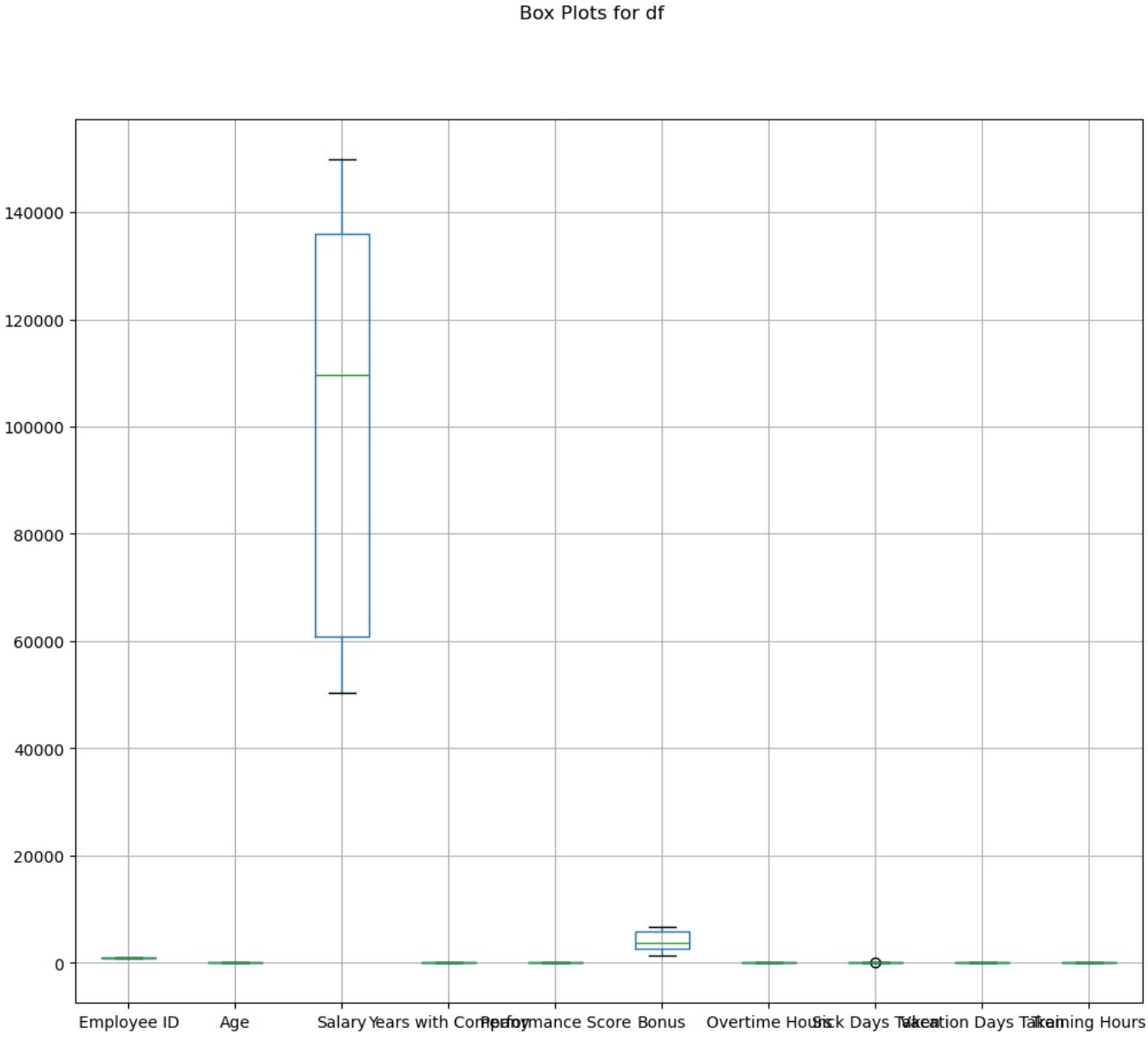
In [35]:
```
# Assuming df1 is your second DataFrame
df1.hist(figsize=(12, 10), bins=15, grid=False)

# Display the plots for df1
plt.suptitle('Histograms for df1')
plt.show()
```



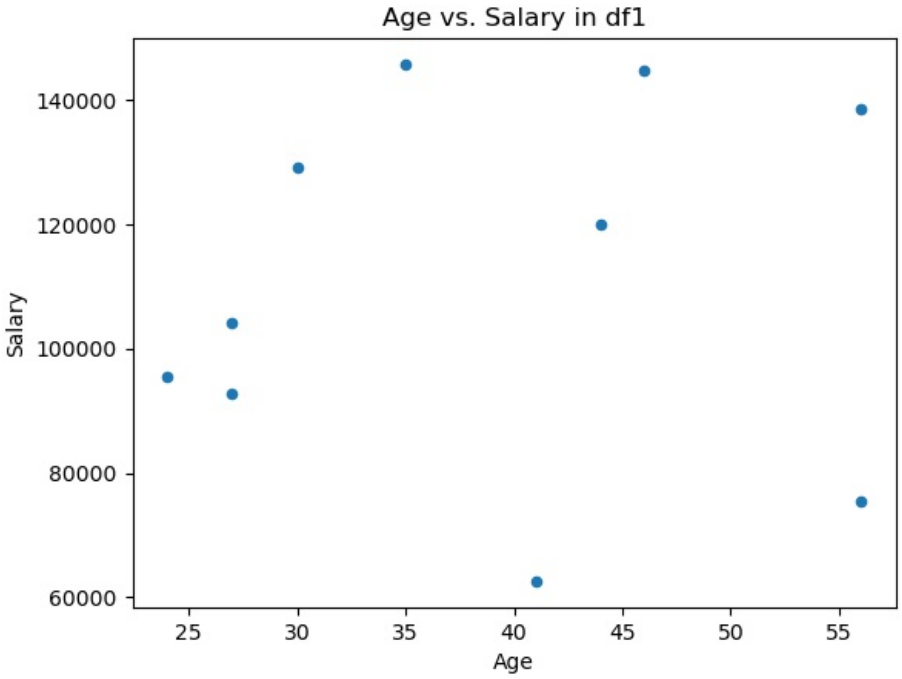
In [36]:
```
# Box plots for df1
df1.boxplot(figsize=(12, 10))
plt.suptitle('Box Plots for df1')
plt.show()
```



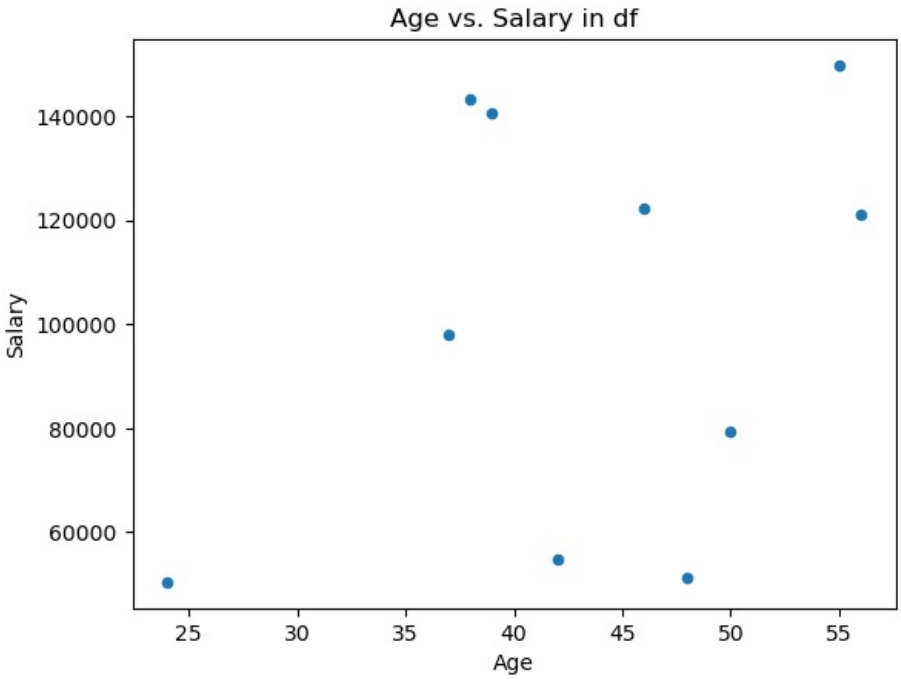
In [37]:
```
# Box plots for df1
df.boxplot(figsize=(12, 10))
plt.suptitle('Box Plots for df')
plt.show()
```



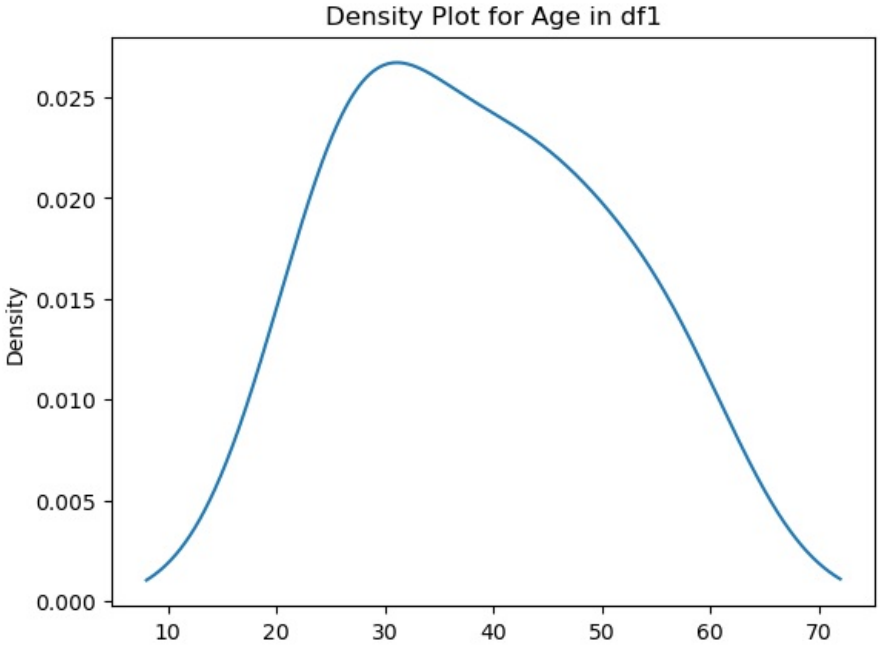
In [44]:
```
# Scatter plot between 'Age' and 'Salary' in df1
df1.plot.scatter(x='Age', y='Salary', title='Age vs. Salary in df1')
plt.show()
```



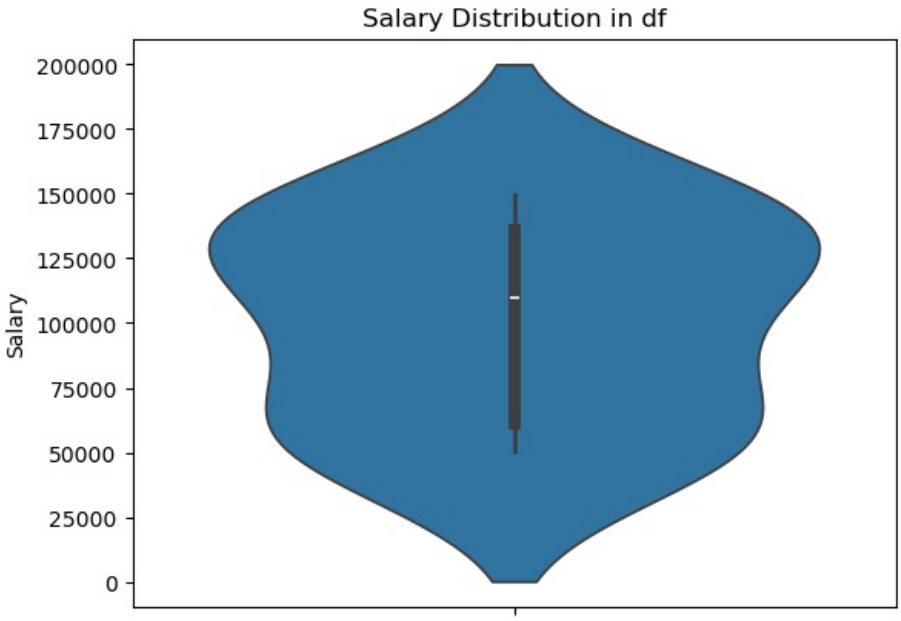
In [45]:
```
# Scatter plot between 'Age' and 'Salary' in df
df.plot.scatter(x='Age', y='Salary', title='Age vs. Salary in df')
plt.show()
```



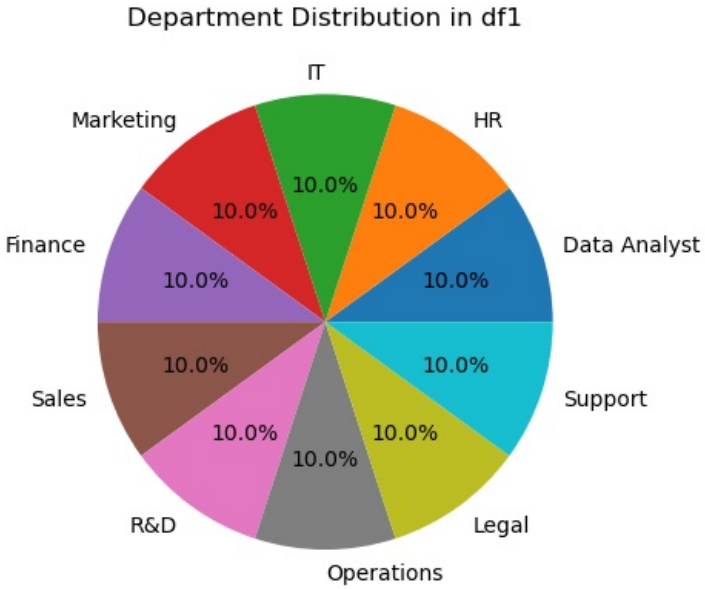
In [48]:
```
# Density plot for Age distribution in df1
df1['Age'].plot(kind='density', title='Density Plot for Age in df1')
plt.show()
```



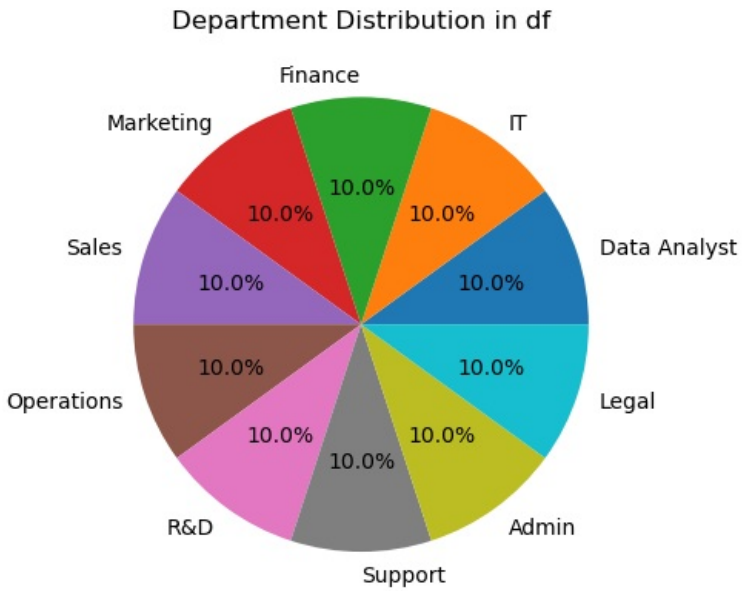
In [51]:
```
import seaborn as sns

# Violin plot for Salary distribution in df
sns.violinplot(y='Salary', data=df)
plt.title('Salary Distribution in df')
plt.show()
```



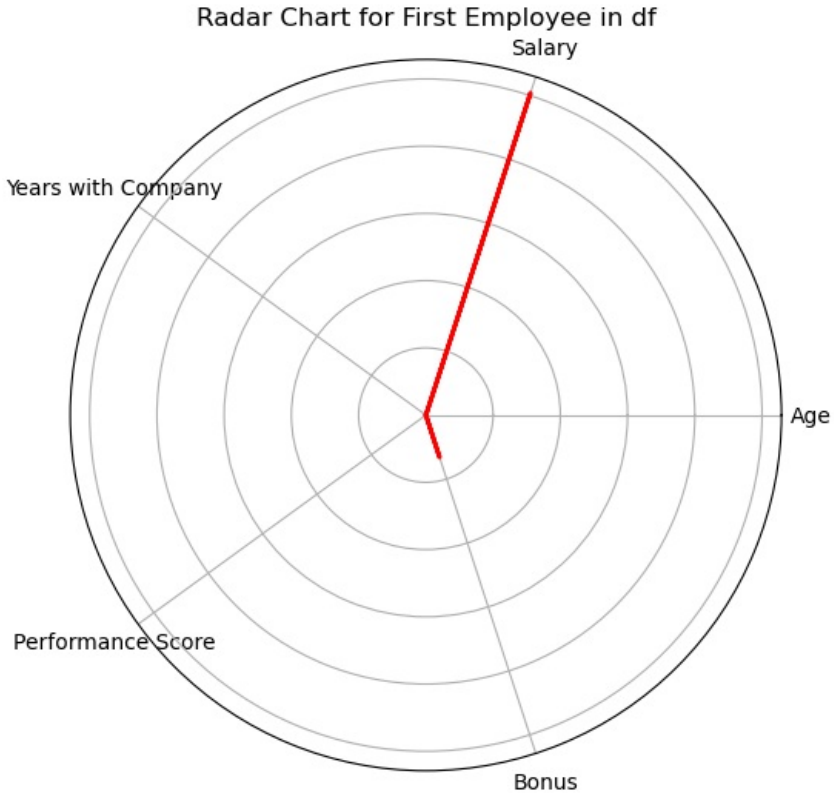
In [52]:
```
# Pie chart for department distribution in df1
df1['Department'].value_counts().plot(kind='pie', autopct='%1.1f%%', title='Department Distribution in df1')
plt.ylabel('')  # Remove the y-label for a cleaner look
plt.show()
```



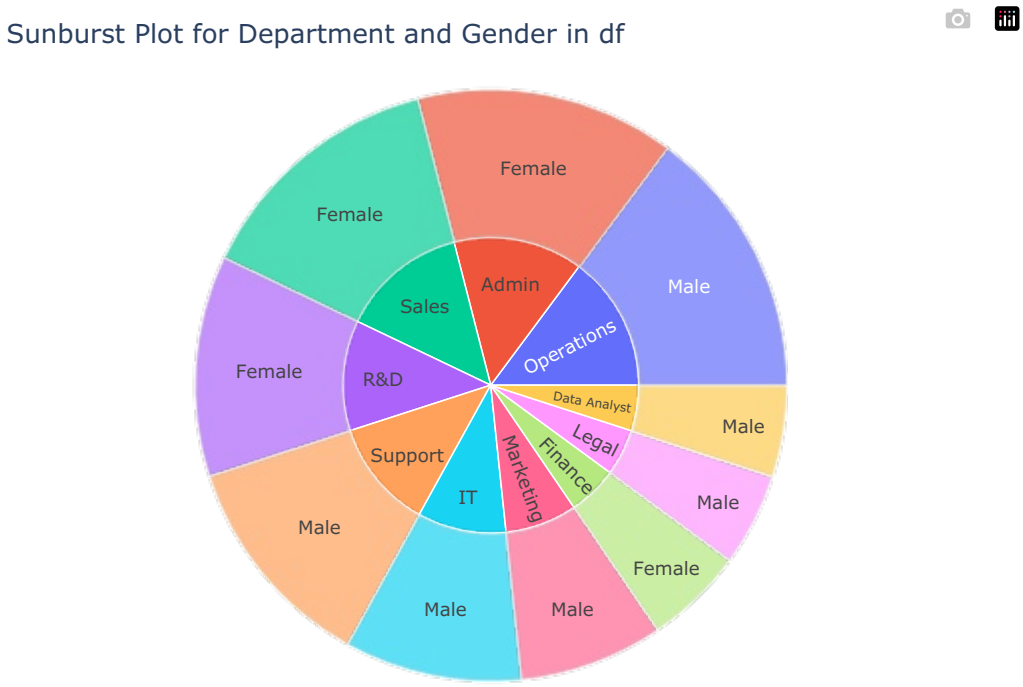
In [53]:
```
# Pie chart for department distribution in df
df['Department'].value_counts().plot(kind='pie', autopct='%1.1f%%', title='Department Distribution in df')
plt.ylabel('')  # Remove the y-label for a cleaner look
plt.show()
```



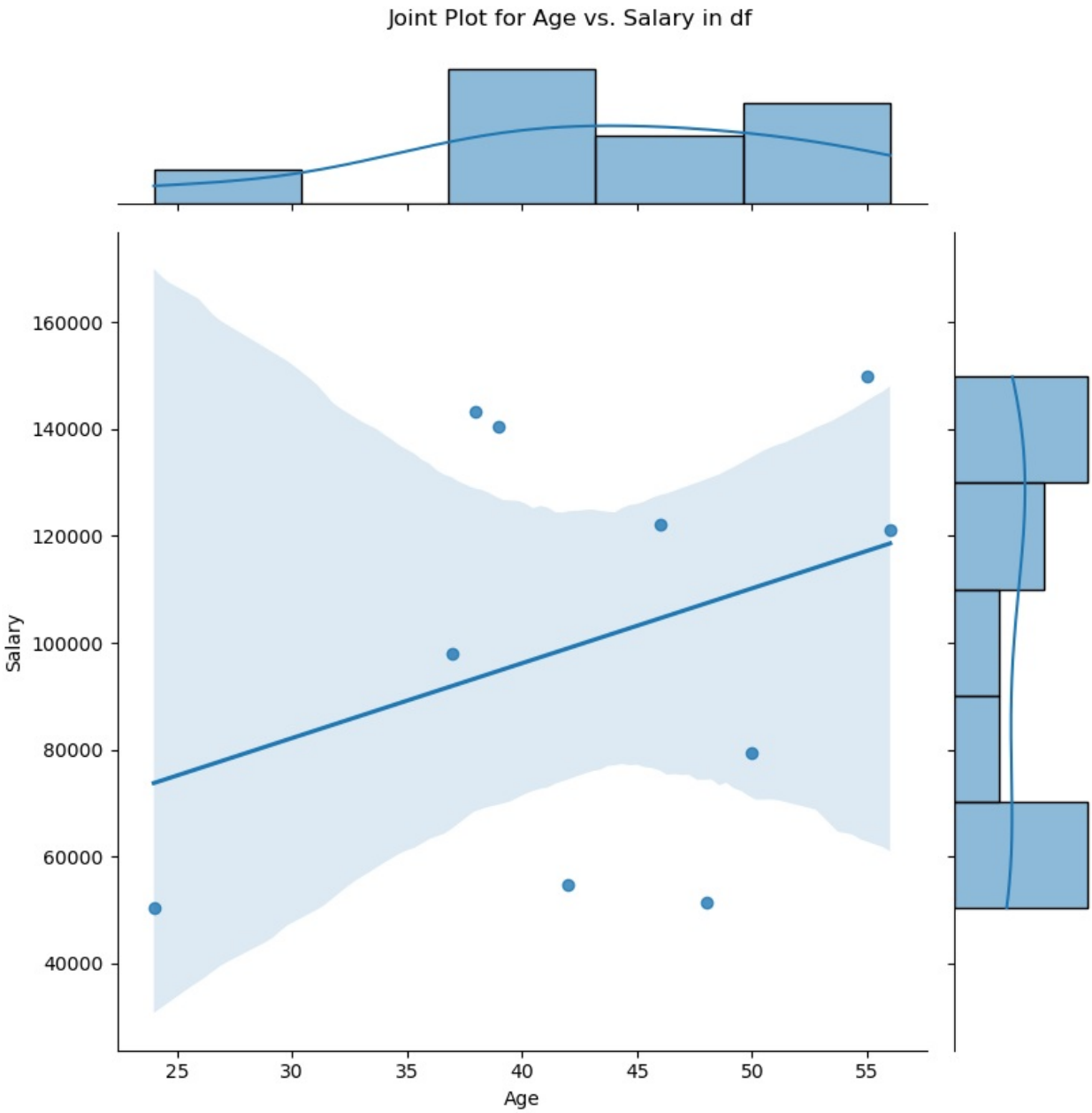
In [68]:
```
import numpy as np

# Radar chart for comparing different metrics for the first employee in df
categories = ['Age', 'Salary', 'Years with Company', 'Performance Score', 'Bonus']
values = df.loc[0, categories].values.flatten().tolist()

# Adding the first value to the end of the list to close the radar chart
values += values[:1]
angles = np.linspace(0, 2 * np.pi, len(categories), endpoint=False).tolist()
angles += angles[:1]

fig, ax = plt.subplots(figsize=(6, 6), subplot_kw=dict(polar=True))
ax.fill(angles, values, color='red', alpha=0.25)
ax.plot(angles, values, color='red', linewidth=2)
ax.set_yticklabels([])
ax.set_xticks(angles[:-1])
ax.set_xticklabels(categories)
plt.title('Radar Chart for First Employee in df')
plt.show()
```



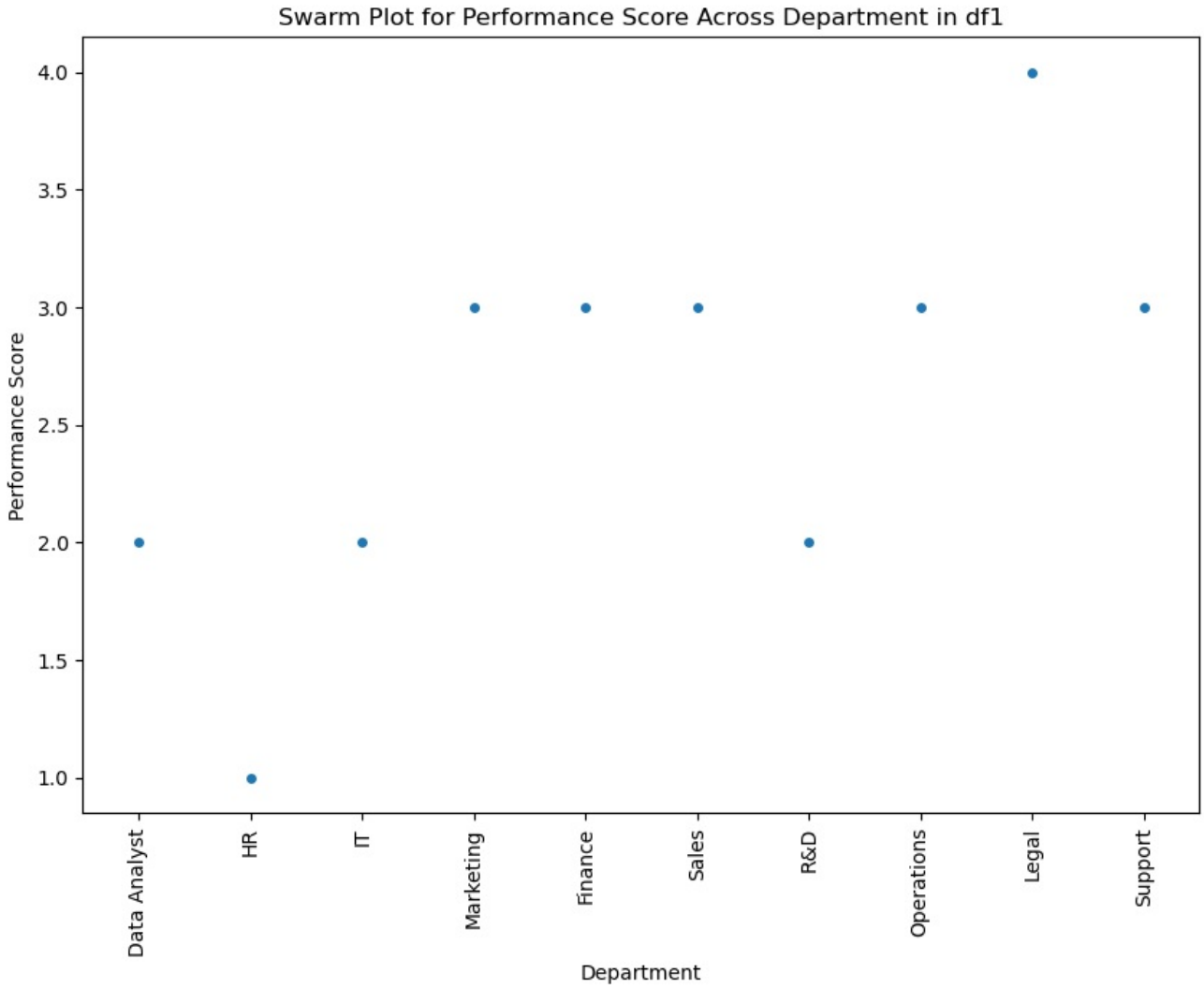
In [69]:
```
import plotly.express as px

# Sunburst plot showing hierarchy of Department and Gender in df
fig = px.sunburst(df, path=['Department', 'Gender'], values='Salary')
fig.update_layout(title='Sunburst Plot for Department and Gender in df')
fig.show()
```



In [72]:
```
# Joint plot for Age vs. Salary in df
sns.jointplot(x='Age', y='Salary', data=df, kind='reg', height=8)
plt.suptitle('Joint Plot for Age vs. Salary in df', y=1.03)
plt.show()
```



In [73]:
```
# Swarm plot for Performance Score across Department in df1
plt.figure(figsize=(10, 7))
sns.swarmplot(x='Department', y='Performance Score', data=df1)
plt.title('Swarm Plot for Performance Score Across Department in df1')
plt.xticks(rotation=90)
plt.show()
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Merge, Concatenate, join and more

In [1]:
```
import pandas as pd  # For data manipulation and analysis
import numpy as np  # For numerical computations
import matplotlib.pyplot as plt  # For creating static, animated, and interactive visualizations
import seaborn as sns  # For statistical data visualization based on Matplotlib
import scipy  # For scientific and technical computing (including optimization, integration, and statistics)
```

In [2]:
```
# Creating realistic data for employees
data = {
    'Employee ID': np.arange(1001, 1011),
    'Employee Name': ['Satender Kumar', 'data 1', 'Jane Smith', 'Robert Brown', 'Emily Davis', 'Michael Wilson'
    'Department': ['Data Analyst', 'IT', 'Finance', 'Marketing', 'Sales', 'Operations', 'R&D', 'Support', 'Admi
    'Age': [24, np.random.randint(25, 60), np.random.randint(25, 60), np.random.randint(25, 60), np.random.rand
    'Location': ['London, Canada', 'Toronto', 'London', 'Sydney', 'San Francisco', 'Paris', 'Berlin', 'Tokyo',
    'Salary': np.random.randint(50000, 150000, size=10),
    'Years with Company': np.random.randint(1, 15, size=10),
    'Position': ['Data Analyst', 'Developer', 'Analyst', 'Designer', 'Consultant', 'Engineer', 'Scientist', 'Su
    'Performance Score': np.random.randint(1, 5, size=10),
    'Bonus': np.random.randint(1000, 10000, size=10),
    'Gender': ['Male', 'Male', 'Female', 'Male', 'Female', 'Male', 'Female', 'Male', 'Female', 'Male'],
    'Marital Status': ['Single', 'Single', 'Married', 'Single', 'Single', 'Married', 'Married', 'Single', 'Marr
    'Education': ['Bachelor', 'Master', 'PhD', 'Bachelor', 'Master', 'PhD', 'Bachelor', 'Master', 'Bachelor', '
    'Hire Date': pd.to_datetime(['2019-06-12', '2015-07-23', '2012-09-05', '2018-11-30', '2013-05-19', '2019-02
    'Overtime Hours': np.random.randint(0, 20, size=10),
    'Sick Days Taken': np.random.randint(0, 10, size=10),
    'Vacation Days Taken': np.random.randint(5, 20, size=10),
    'Training Hours': np.random.randint(10, 50, size=10),
    'Certifications': ['Yes', 'No', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'No', 'Yes'],
    'Supervisor': ['Anna Smith', 'Brian Adams', 'Clara Jones', 'Daniel Martin', 'Eva Rodriguez', 'Frank Bell',
}

# Creating the DataFrame
df = pd.DataFrame(data)
```

In [4]:
```
# Display the DataFrame
print(df)
```

```
   Employee ID   Employee Name    Department  Age        Location  Salary  \
0         1001  Satender Kumar  Data Analyst   24  London, Canada   62989
1         1002          data 1            IT   46         Toronto   96061
2         1003      Jane Smith       Finance   48          London   60507
3         1004    Robert Brown     Marketing   32          Sydney   71727
4         1005     Emily Davis         Sales   40   San Francisco   81098
5         1006  Michael Wilson    Operations   45           Paris   60924
6         1007    Sarah Taylor           R&D   58          Berlin  116105
7         1008       David Lee       Support   28           Tokyo  138939
8         1009   Laura Johnson         Admin   38           Dubai   93249
9         1010     James White         Legal   46       Singapore  141721

   Years with Company         Position  Performance Score  Bonus  Gender  \
0                  10     Data Analyst                  4   6419    Male
1                   5        Developer                  4   4746    Male
2                   1          Analyst                  1   1447  Female
3                   9         Designer                  3   7196    Male
4                  13       Consultant                  3   2218  Female
5                   6         Engineer                  2   7973    Male
6                  13        Scientist                  4   7565  Female
7                   9    Support Agent                  3   1800    Male
8                   7  Admin Assistant                  4   1189  Female
9                   2           Lawyer                  2   1654    Male

  Marital Status Education  Hire Date  Overtime Hours  Sick Days Taken  \
0         Single  Bachelor 2019-06-12              10                2
1         Single    Master 2015-07-23              15                7
2        Married       PhD 2012-09-05               0                5
3         Single  Bachelor 2018-11-30              18                3
4         Single    Master 2013-05-19               1                7
5        Married       PhD 2019-02-14               0                1
6        Married  Bachelor 2020-08-21               3                0
7         Single    Master 2016-06-03               6                0
8        Married  Bachelor 2014-01-28              15                1
9         Single    Master 2017-03-15               3                7

   Vacation Days Taken  Training Hours Certifications     Supervisor
0                   19              15            Yes     Anna Smith
1                   14              31             No    Brian Adams
2                    7              37            Yes    Clara Jones
3                   11              12             No  Daniel Martin
4                   17              30            Yes  Eva Rodriguez
5                   16              24             No     Frank Bell
6                    7              49            Yes    Grace Moore
7                    6              10            Yes   Hannah Lewis
8                   11              38             No     Ivan Scott
9                   19              40            Yes    Jake Miller
```

In [6]:
```
df.head()
```

Out[6]:

	Employee ID	Employee Name	Department	Age	Location	Salary	Years with Company	Position	Performance Score	Bonus	Gender	Marital Status	Educati
0	1001	Satender Kumar	Data Analyst	24	London, Canada	62989	10	Data Analyst	4	6419	Male	Single	Bache
1	1002	data 1	IT	46	Toronto	96061	5	Developer	4	4746	Male	Single	Mas
2	1003	Jane Smith	Finance	48	London	60507	1	Analyst	1	1447	Female	Married	Pl
3	1004	Robert Brown	Marketing	32	Sydney	71727	9	Designer	3	7196	Male	Single	Bache
4	1005	Emily Davis	Sales	40	San Francisco	81098	13	Consultant	3	2218	Female	Single	Mas

In [10]:
```
df.describe()
```

Out[10]:

	Employee ID	Age	Salary	Years with Company	Performance Score	Bonus	Hire Date	Overtime Hours	Sick Days Taken	Vacation Days Taken	Tr
count	10.00000	10.000000	10.000000	10.000000	10.000000	10.000000	10	10.000000	10.000000	10.000000	10.(
mean	1005.50000	40.500000	92332.000000	7.500000	3.000000	4220.700000	2016-09-28 09:36:00	7.100000	3.300000	12.700000	28.(
min	1001.00000	24.000000	60507.000000	1.000000	1.000000	1189.000000	2012-09-05 00:00:00	0.000000	0.000000	6.000000	10.(
25%	1003.25000	33.500000	65173.500000	5.250000	2.250000	1690.500000	2014-06-12 06:00:00	1.500000	1.000000	8.000000	17.:
50%	1005.50000	42.500000	87173.500000	8.000000	3.000000	3482.000000	2016-10-23 12:00:00	4.500000	2.500000	12.500000	30.:
75%	1007.75000	46.000000	111094.000000	9.750000	4.000000	7001.750000	2019-01-26 00:00:00	13.750000	6.500000	16.750000	37.7
max	1010.00000	58.000000	141721.000000	13.000000	4.000000	7973.000000	2020-08-21 00:00:00	18.000000	7.000000	19.000000	49.(
std	3.02765	10.276727	30916.484542	4.116363	1.054093	2839.564366	NaN	6.871034	2.945807	5.012207	13.(

In [12]:
```
df.isnull()
```

Out[12]:

	Employee ID	Employee Name	Department	Age	Location	Salary	Years with Company	Position	Performance Score	Bonus	Gender	Marital Status	Educatio
0	False	False	False	False	False	False	False	False	False	False	False	False	Fals
1	False	False	False	False	False	False	False	False	False	False	False	False	Fals
2	False	False	False	False	False	False	False	False	False	False	False	False	Fals
3	False	False	False	False	False	False	False	False	False	False	False	False	Fals
4	False	False	False	False	False	False	False	False	False	False	False	False	Fals
5	False	False	False	False	False	False	False	False	False	False	False	False	Fals
6	False	False	False	False	False	False	False	False	False	False	False	False	Fals
7	False	False	False	False	False	False	False	False	False	False	False	False	Fals
8	False	False	False	False	False	False	False	False	False	False	False	False	Fals
9	False	False	False	False	False	False	False	False	False	False	False	False	Fals

In [14]:
```
df.notnull()
```

Out[14]:

	Employee ID	Employee Name	Department	Age	Location	Salary	Years with Company	Position	Performance Score	Bonus	Gender	Marital Status	Education
0	True	True	True	True	True	True	True	True	True	True	True	True	True
1	True	True	True	True	True	True	True	True	True	True	True	True	True
2	True	True	True	True	True	True	True	True	True	True	True	True	True
3	True	True	True	True	True	True	True	True	True	True	True	True	True
4	True	True	True	True	True	True	True	True	True	True	True	True	True
5	True	True	True	True	True	True	True	True	True	True	True	True	True
6	True	True	True	True	True	True	True	True	True	True	True	True	True
7	True	True	True	True	True	True	True	True	True	True	True	True	True
8	True	True	True	True	True	True	True	True	True	True	True	True	True
9	True	True	True	True	True	True	True	True	True	True	True	True	True

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

Out[16]:
```
Employee ID             10
Employee Name           10
Department              10
Age                     10
Location                10
Salary                  10
Years with Company      10
Position                10
Performance Score       10
Bonus                   10
Gender                  10
Marital Status          10
Education               10
Hire Date               10
Overtime Hours          10
Sick Days Taken         10
Vacation Days Taken     10
Training Hours          10
Certifications          10
Supervisor              10
dtype: int64
```

In [19]:
```
df.head(20
       )
```

Out[19]:

	Employee ID	Employee Name	Department	Age	Location	Salary	Years with Company	Position	Performance Score	Bonus	Gender	Marital Status	Educat
0	1001	Satender Kumar	Data Analyst	24	London, Canada	62989	10	Data Analyst	4	6419	Male	Single	Bach
1	1002	data 1	IT	46	Toronto	96061	5	Developer	4	4746	Male	Single	Ma
2	1003	Jane Smith	Finance	48	London	60507	1	Analyst	1	1447	Female	Married	
3	1004	Robert Brown	Marketing	32	Sydney	71727	9	Designer	3	7196	Male	Single	Bach
4	1005	Emily Davis	Sales	40	San Francisco	81098	13	Consultant	3	2218	Female	Single	Ma
5	1006	Michael Wilson	Operations	45	Paris	60924	6	Engineer	2	7973	Male	Married	
6	1007	Sarah Taylor	R&D	58	Berlin	116105	13	Scientist	4	7565	Female	Married	Bach
7	1008	David Lee	Support	28	Tokyo	138939	9	Support Agent	3	1800	Male	Single	Ma
8	1009	Laura Johnson	Admin	38	Dubai	93249	7	Admin Assistant	4	1189	Female	Married	Bach
9	1010	James White	Legal	46	Singapore	141721	2	Lawyer	2	1654	Male	Single	Ma

In [21]:
```
# Creating realistic data for a second set of employees
data1 = {
    'Employee ID': np.arange(1011, 1021),
    'Employee Name': ['Satender Kumar', 'data 1', 'Chris Evans', 'Natalie Portman', 'Tom Holland', 'Emma Watson
    'Department': ['Data Analyst', 'HR', 'IT', 'Marketing', 'Finance', 'Sales', 'R&D', 'Operations', 'Legal', '
    'Age': [24, np.random.randint(25, 60), np.random.randint(25, 60), np.random.randint(25, 60), np.random.randi
    'Location': ['London, Canada', 'Los Angeles', 'New York', 'Chicago', 'Houston', 'Phoenix', 'Philadelphia',
    'Salary': np.random.randint(60000, 160000, size=10),
    'Years with Company': np.random.randint(1, 20, size=10),
    'Position': ['Data Analyst', 'HR Manager', 'IT Specialist', 'Marketing Coordinator', 'Financial Analyst', '
    'Performance Score': np.random.randint(1, 5, size=10),
    'Bonus': np.random.randint(2000, 12000, size=10),
    'Gender': ['Male', 'Male', 'Female', 'Female', 'Male', 'Female', 'Male', 'Female', 'Male', 'Male'],
    'Marital Status': ['Single', 'Married', 'Single', 'Single', 'Married', 'Single', 'Single', 'Married', 'Sing
    'Education': ['Master', 'Bachelor', 'Master', 'PhD', 'Bachelor', 'Master', 'PhD', 'Bachelor', 'Master', 'Phl
    'Hire Date': pd.to_datetime(['2018-07-15', '2014-03-22', '2011-10-12', '2017-04-17', '2015-09-23', '2016-11
    'Overtime Hours': np.random.randint(0, 25, size=10),
    'Sick Days Taken': np.random.randint(0, 8, size=10),
    'Vacation Days Taken': np.random.randint(7, 22, size=10),
    'Training Hours': np.random.randint(15, 55, size=10),
    'Certifications': ['Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'No'],
    'Supervisor': ['John Smith', 'Michael Johnson', 'Patricia Williams', 'Linda Brown', 'Barbara Jones', 'Eliza
}

# Creating the second DataFrame
df1 = pd.DataFrame(data1)
```

In [23]:
```
# Display the DataFrame
print(df1)
```

```
   Employee ID       Employee Name    Department  Age        Location  Salary  \
0         1011      Satender Kumar  Data Analyst   24  London, Canada   75719
1         1012              data 1            HR   40     Los Angeles   96413
2         1013         Chris Evans            IT   37        New York   74042
3         1014     Natalie Portman     Marketing   49         Chicago  108467
4         1015         Tom Holland       Finance   27         Houston   65822
5         1016         Emma Watson         Sales   56         Phoenix  114854
6         1017    Daniel Radcliffe           R&D   33    Philadelphia  117933
7         1018  Scarlett Johansson    Operations   41     San Antonio   87342
8         1019   Robert Downey Jr.         Legal   51       San Diego   72775
9         1020        Mark Ruffalo       Support   42          Dallas  142745

   Years with Company              Position  Performance Score  Bonus  \
0                  12          Data Analyst                  2   3187
1                  12            HR Manager                  1  10993
2                  14         IT Specialist                  4  10834
3                  11  Marketing Coordinator                  4   2760
4                  12      Financial Analyst                  2   6263
5                   5          Sales Manager                  3  10287
6                  17     Research Scientist                  3   8290
7                   8     Operations Manager                  3  10047
8                   6          Legal Advisor                  2   6510
9                  10     Support Specialist                  1   5533

   Gender Marital Status Education  Hire Date  Overtime Hours  \
0    Male         Single    Master 2018-07-15              17
1    Male        Married  Bachelor 2014-03-22               8
2  Female         Single    Master 2011-10-12              12
3  Female         Single       PhD 2017-04-17              10
4    Male        Married  Bachelor 2015-09-23              13
5  Female         Single    Master 2016-11-01              15
6    Male         Single       PhD 2019-05-11               8
7  Female        Married  Bachelor 2020-07-08              15
8    Male         Single    Master 2013-08-19               7
9    Male        Married       PhD 2012-01-09               2

   Sick Days Taken  Vacation Days Taken  Training Hours Certifications  \
0                5                   12             21            Yes
1                3                   18             41            Yes
2                4                    8             22             No
3                7                   20             42            Yes
4                6                   11             35             No
5                0                   20             35            Yes
6                1                   15             48             No
7                7                    9             50            Yes
8                7                   20             35            Yes
9                4                   17             15             No

         Supervisor
0        John Smith
1   Michael Johnson
2  Patricia Williams
3       Linda Brown
4     Barbara Jones
5   Elizabeth Garcia
6    Susan Martinez
7  Jessica Hernandez
8       Sarah Lopez
9      Karen Wilson
```

In [25]:
```
df1.head()
```

Out[25]:

	Employee ID	Employee Name	Department	Age	Location	Salary	Years with Company	Position	Performance Score	Bonus	Gender	Marital Status	Educa
0	1011	Satender Kumar	Data Analyst	24	London, Canada	75719	12	Data Analyst	2	3187	Male	Single	Ma
1	1012	data 1	HR	40	Los Angeles	96413	12	HR Manager	1	10993	Male	Married	Bach
2	1013	Chris Evans	IT	37	New York	74042	14	IT Specialist	4	10834	Female	Single	Ma
3	1014	Natalie Portman	Marketing	49	Chicago	108467	11	Marketing Coordinator	4	2760	Female	Single	I
4	1015	Tom Holland	Finance	27	Houston	65822	12	Financial Analyst	2	6263	Male	Married	Bach

In [27]:
```
df.describe()
```

Out[27]:

	Employee ID	Age	Salary	Years with Company	Performance Score	Bonus	Hire Date	Overtime Hours	Sick Days Taken	Vacation Days Taken	Tr
count	10.00000	10.000000	10.000000	10.000000	10.000000	10.000000	10	10.000000	10.000000	10.000000	10.(
mean	1005.50000	40.500000	92332.000000	7.500000	3.000000	4220.700000	2016-09-28 09:36:00	7.100000	3.300000	12.700000	28.(
min	1001.00000	24.000000	60507.000000	1.000000	1.000000	1189.000000	2012-09-05 00:00:00	0.000000	0.000000	6.000000	10.(
25%	1003.25000	33.500000	65173.500000	5.250000	2.250000	1690.500000	2014-06-12 06:00:00	1.500000	1.000000	8.000000	17.:
50%	1005.50000	42.500000	87173.500000	8.000000	3.000000	3482.000000	2016-10-23 12:00:00	4.500000	2.500000	12.500000	30.:
75%	1007.75000	46.000000	111094.000000	9.750000	4.000000	7001.750000	2019-01-26 00:00:00	13.750000	6.500000	16.750000	37.7
max	1010.00000	58.000000	141721.000000	13.000000	4.000000	7973.000000	2020-08-21 00:00:00	18.000000	7.000000	19.000000	49.(
std	3.02765	10.276727	30916.484542	4.116363	1.054093	2839.564366	NaN	6.871034	2.945807	5.012207	13.(

In [28]:
```
df1.notnull()
```

Out[28]:

	Employee ID	Employee Name	Department	Age	Location	Salary	Years with Company	Position	Performance Score	Bonus	Gender	Marital Status	Education
0	True	True	True	True	True	True	True	True	True	True	True	True	True
1	True	True	True	True	True	True	True	True	True	True	True	True	True
2	True	True	True	True	True	True	True	True	True	True	True	True	True
3	True	True	True	True	True	True	True	True	True	True	True	True	True
4	True	True	True	True	True	True	True	True	True	True	True	True	True
5	True	True	True	True	True	True	True	True	True	True	True	True	True
6	True	True	True	True	True	True	True	True	True	True	True	True	True
7	True	True	True	True	True	True	True	True	True	True	True	True	True
8	True	True	True	True	True	True	True	True	True	True	True	True	True
9	True	True	True	True	True	True	True	True	True	True	True	True	True

In [29]:
```
df1.isnull()
```

Out[29]:

	Employee ID	Employee Name	Department	Age	Location	Salary	Years with Company	Position	Performance Score	Bonus	Gender	Marital Status	Educatio
0	False	False	False	False	False	False	False	False	False	False	False	False	Fals
1	False	False	False	False	False	False	False	False	False	False	False	False	Fals
2	False	False	False	False	False	False	False	False	False	False	False	False	Fals
3	False	False	False	False	False	False	False	False	False	False	False	False	Fals
4	False	False	False	False	False	False	False	False	False	False	False	False	Fals
5	False	False	False	False	False	False	False	False	False	False	False	False	Fals
6	False	False	False	False	False	False	False	False	False	False	False	False	Fals
7	False	False	False	False	False	False	False	False	False	False	False	False	Fals
8	False	False	False	False	False	False	False	False	False	False	False	False	Fals
9	False	False	False	False	False	False	False	False	False	False	False	False	Fals

In [30]:
```
df1.count()
```

Out[30]:
```
Employee ID            10
Employee Name          10
Department             10
Age                    10
Location               10
Salary                 10
Years with Company     10
Position               10
Performance Score      10
Bonus                  10
Gender                 10
Marital Status         10
Education              10
Hire Date              10
Overtime Hours         10
Sick Days Taken        10
Vacation Days Taken    10
Training Hours         10
Certifications         10
Supervisor             10
dtype: int64
```

In [ ]:
```
df1.sum()
```

In [33]:
```
print(df1)
```

```
   Employee ID       Employee Name    Department  Age        Location  Salary  \
0         1011      Satender Kumar  Data Analyst   24  London, Canada   75719
1         1012              data 1            HR   40     Los Angeles   96413
2         1013         Chris Evans            IT   37        New York   74042
3         1014     Natalie Portman     Marketing   49         Chicago  108467
4         1015         Tom Holland       Finance   27         Houston   65822
5         1016         Emma Watson         Sales   56         Phoenix  114854
6         1017    Daniel Radcliffe           R&D   33    Philadelphia  117933
7         1018  Scarlett Johansson    Operations   41     San Antonio   87342
8         1019   Robert Downey Jr.         Legal   51       San Diego   72775
9         1020        Mark Ruffalo       Support   42          Dallas  142745

   Years with Company              Position  Performance Score  Bonus  \
0                  12          Data Analyst                  2   3187
1                  12            HR Manager                  1  10993
2                  14        IT Specialist                  4  10834
3                  11  Marketing Coordinator                  4   2760
4                  12     Financial Analyst                  2   6263
5                   5         Sales Manager                  3  10287
6                  17     Research Scientist                  3   8290
7                   8     Operations Manager                  3  10047
8                   6          Legal Advisor                  2   6510
9                  10     Support Specialist                  1   5533

   Gender Marital Status Education  Hire Date  Overtime Hours  \
0    Male         Single    Master 2018-07-15              17
1    Male        Married  Bachelor 2014-03-22               8
2  Female         Single    Master 2011-10-12              12
3  Female         Single       PhD 2017-04-17              10
4    Male        Married  Bachelor 2015-09-23              13
5  Female         Single    Master 2016-11-01              15
6    Male         Single       PhD 2019-05-11               8
7  Female        Married  Bachelor 2020-07-08              15
8    Male         Single    Master 2013-08-19               7
9    Male        Married       PhD 2012-01-09               2

   Sick Days Taken  Vacation Days Taken  Training Hours Certifications  \
0                5                   12              21            Yes
1                3                   18              41            Yes
2                4                    8              22             No
3                7                   20              42            Yes
4                6                   11              35             No
5                0                   20              35            Yes
6                1                   15              48             No
7                7                    9              50            Yes
8                7                   20              35            Yes
9                4                   17              15             No

         Supervisor
0        John Smith
1   Michael Johnson
2  Patricia Williams
3       Linda Brown
4     Barbara Jones
5   Elizabeth Garcia
6    Susan Martinez
7  Jessica Hernandez
8       Sarah Lopez
9      Karen Wilson
```

In [36]:

```python
import pandas as pd

# Assuming df and df1 are your DataFrames from the previous examples

# Merging the two DataFrames on the 'Employee ID' column
merged_df = pd.merge(df, df1, on='Employee ID', suffixes=('_df', '_df1'))

# Display the merged DataFrame
print(merged_df)
```

```
Empty DataFrame
Columns: [Employee ID, Employee Name_df, Department_df, Age_df, Location_df, Salary_df, Years with Company_df, Position_df, Performance Score_df, Bonus_df, Gender_df, Marital Status_df, Education_df, Hire Date_df, Overtime Hours_df, Sick Days Taken_df, Vacation Days Taken_df, Training Hours_df, Certifications_df, Supervisor_df, Employee Name_df1, Department_df1, Age_df1, Location_df1, Salary_df1, Years with Company_df1, Position_df1, Performance Score_df1, Bonus_df1, Gender_df1, Marital Status_df1, Education_df1, Hire Date_df1, Overtime Hours_df1, Sick Days Taken_df1, Vacation Days Taken_df1, Training Hours_df1, Certifications_df1, Supervisor_df1]
Index: []

[0 rows x 39 columns]
```

In [39]:

```python
merged_df = pd.merge(df, df1, on='Employee ID', suffixes=('_df', '_df1'), how='outer')
print(merged_df)
```

	Employee ID	Employee Name_df	Department_df	Age_df	Location_df
0	1001	Satender Kumar	Data Analyst	24.0	London, Canada
1	1002	data 1	IT	46.0	Toronto
2	1003	Jane Smith	Finance	48.0	London
3	1004	Robert Brown	Marketing	32.0	Sydney
4	1005	Emily Davis	Sales	40.0	San Francisco
5	1006	Michael Wilson	Operations	45.0	Paris
6	1007	Sarah Taylor	R&D	58.0	Berlin
7	1008	David Lee	Support	28.0	Tokyo
8	1009	Laura Johnson	Admin	38.0	Dubai
9	1010	James White	Legal	46.0	Singapore
10	1011	NaN	NaN	NaN	NaN
11	1012	NaN	NaN	NaN	NaN
12	1013	NaN	NaN	NaN	NaN
13	1014	NaN	NaN	NaN	NaN
14	1015	NaN	NaN	NaN	NaN
15	1016	NaN	NaN	NaN	NaN
16	1017	NaN	NaN	NaN	NaN
17	1018	NaN	NaN	NaN	NaN
18	1019	NaN	NaN	NaN	NaN
19	1020	NaN	NaN	NaN	NaN

	Salary_df	Years with Company_df	Position_df	Performance Score_df
0	62989.0	10.0	Data Analyst	4.0
1	96061.0	5.0	Developer	4.0
2	60507.0	1.0	Analyst	1.0
3	71727.0	9.0	Designer	3.0
4	81098.0	13.0	Consultant	3.0
5	60924.0	6.0	Engineer	2.0
6	116105.0	13.0	Scientist	4.0
7	138939.0	9.0	Support Agent	3.0
8	93249.0	7.0	Admin Assistant	4.0
9	141721.0	2.0	Lawyer	2.0
10	NaN	NaN	NaN	NaN
11	NaN	NaN	NaN	NaN
12	NaN	NaN	NaN	NaN
13	NaN	NaN	NaN	NaN
14	NaN	NaN	NaN	NaN
15	NaN	NaN	NaN	NaN
16	NaN	NaN	NaN	NaN
17	NaN	NaN	NaN	NaN
18	NaN	NaN	NaN	NaN
19	NaN	NaN	NaN	NaN

	Bonus_df	...	Gender_df1	Marital Status_df1	Education_df1	Hire Date_df1
0	6419.0	...	NaN	NaN	NaN	NaT
1	4746.0	...	NaN	NaN	NaN	NaT
2	1447.0	...	NaN	NaN	NaN	NaT
3	7196.0	...	NaN	NaN	NaN	NaT
4	2218.0	...	NaN	NaN	NaN	NaT
5	7973.0	...	NaN	NaN	NaN	NaT
6	7565.0	...	NaN	NaN	NaN	NaT
7	1800.0	...	NaN	NaN	NaN	NaT
8	1189.0	...	NaN	NaN	NaN	NaT
9	1654.0	...	NaN	NaN	NaN	NaT
10	NaN	...	Male	Single	Master	2018-07-15
11	NaN	...	Male	Married	Bachelor	2014-03-22
12	NaN	...	Female	Single	Master	2011-10-12
13	NaN	...	Female	Single	PhD	2017-04-17
14	NaN	...	Male	Married	Bachelor	2015-09-23
15	NaN	...	Female	Single	Master	2016-11-01
16	NaN	...	Male	Single	PhD	2019-05-11
17	NaN	...	Female	Married	Bachelor	2020-07-08
18	NaN	...	Male	Single	Master	2013-08-19
19	NaN	...	Male	Married	PhD	2012-01-09

	Overtime Hours_df1	Sick Days Taken_df1	Vacation Days Taken_df1
0	NaN	NaN	NaN
1	NaN	NaN	NaN
2	NaN	NaN	NaN
3	NaN	NaN	NaN
4	NaN	NaN	NaN
5	NaN	NaN	NaN
6	NaN	NaN	NaN
7	NaN	NaN	NaN
8	NaN	NaN	NaN
9	NaN	NaN	NaN
10	17.0	5.0	12.0
11	8.0	3.0	18.0
12	12.0	4.0	8.0
13	10.0	7.0	20.0
14	13.0	6.0	11.0
15	15.0	0.0	20.0

```
16                8.0                 1.0                    15.0
17               15.0                 7.0                     9.0
18                7.0                 7.0                    20.0
19                2.0                 4.0                    17.0

    Training Hours_df1 Certifications_df1     Supervisor_df1
0                  NaN                 NaN                NaN
1                  NaN                 NaN                NaN
2                  NaN                 NaN                NaN
3                  NaN                 NaN                NaN
4                  NaN                 NaN                NaN
5                  NaN                 NaN                NaN
6                  NaN                 NaN                NaN
7                  NaN                 NaN                NaN
8                  NaN                 NaN                NaN
9                  NaN                 NaN                NaN
10                21.0                 Yes         John Smith
11                41.0                 Yes    Michael Johnson
12                22.0                  No  Patricia Williams
13                42.0                 Yes        Linda Brown
14                35.0                  No      Barbara Jones
15                35.0                 Yes   Elizabeth Garcia
16                48.0                  No     Susan Martinez
17                50.0                 Yes  Jessica Hernandez
18                35.0                 Yes        Sarah Lopez
19                15.0                  No       Karen Wilson

[20 rows x 39 columns]
```

In [43]:
```
print(df['Employee ID'])
print(df1['Employee ID'])
```

```
0    1001
1    1002
2    1003
3    1004
4    1005
5    1006
6    1007
7    1008
8    1009
9    1010
Name: Employee ID, dtype: int32
0    1011
1    1012
2    1013
3    1014
4    1015
5    1016
6    1017
7    1018
8    1019
9    1020
Name: Employee ID, dtype: int32
```

In [45]:
```
df
```

Out[45]:

	Employee ID	Employee Name	Department	Age	Location	Salary	Years with Company	Position	Performance Score	Bonus	Gender	Marital Status	Educat
0	1001	Satender Kumar	Data Analyst	24	London, Canada	62989	10	Data Analyst	4	6419	Male	Single	Bach
1	1002	data 1	IT	46	Toronto	96061	5	Developer	4	4746	Male	Single	Ma
2	1003	Jane Smith	Finance	48	London	60507	1	Analyst	1	1447	Female	Married	
3	1004	Robert Brown	Marketing	32	Sydney	71727	9	Designer	3	7196	Male	Single	Bach
4	1005	Emily Davis	Sales	40	San Francisco	81098	13	Consultant	3	2218	Female	Single	Ma
5	1006	Michael Wilson	Operations	45	Paris	60924	6	Engineer	2	7973	Male	Married	
6	1007	Sarah Taylor	R&D	58	Berlin	116105	13	Scientist	4	7565	Female	Married	Bach
7	1008	David Lee	Support	28	Tokyo	138939	9	Support Agent	3	1800	Male	Single	Ma
8	1009	Laura Johnson	Admin	38	Dubai	93249	7	Admin Assistant	4	1189	Female	Married	Bach
9	1010	James White	Legal	46	Singapore	141721	2	Lawyer	2	1654	Male	Single	Ma

In [46]:
```
df1
```

Out[46]:

	Employee ID	Employee Name	Department	Age	Location	Salary	Years with Company	Position	Performance Score	Bonus	Gender	Marital Status	Edu
0	1011	Satender Kumar	Data Analyst	24	London, Canada	75719	12	Data Analyst	2	3187	Male	Single	
1	1012	data 1	HR	40	Los Angeles	96413	12	HR Manager	1	10993	Male	Married	Ba
2	1013	Chris Evans	IT	37	New York	74042	14	IT Specialist	4	10834	Female	Single	
3	1014	Natalie Portman	Marketing	49	Chicago	108467	11	Marketing Coordinator	4	2760	Female	Single	
4	1015	Tom Holland	Finance	27	Houston	65822	12	Financial Analyst	2	6263	Male	Married	Ba
5	1016	Emma Watson	Sales	56	Phoenix	114854	5	Sales Manager	3	10287	Female	Single	
6	1017	Daniel Radcliffe	R&D	33	Philadelphia	117933	17	Research Scientist	3	8290	Male	Single	
7	1018	Scarlett Johansson	Operations	41	San Antonio	87342	8	Operations Manager	3	10047	Female	Married	Ba
8	1019	Robert Downey Jr.	Legal	51	San Diego	72775	6	Legal Advisor	2	6510	Male	Single	
9	1020	Mark Ruffalo	Support	42	Dallas	142745	10	Support Specialist	1	5533	Male	Married	

In [50]:
```
print(pd.concat([df,df1]))
```

```
   Employee ID       Employee Name    Department  Age        Location  Salary  \
0         1001      Satender Kumar  Data Analyst   24  London, Canada   62989
1         1002              data 1            IT   46         Toronto   96061
2         1003          Jane Smith       Finance   48          London   60507
3         1004        Robert Brown     Marketing   32          Sydney   71727
4         1005         Emily Davis         Sales   40   San Francisco   81098
5         1006      Michael Wilson    Operations   45           Paris   60924
6         1007        Sarah Taylor           R&D   58          Berlin  116105
7         1008           David Lee       Support   28           Tokyo  138939
8         1009       Laura Johnson         Admin   38           Dubai   93249
9         1010         James White         Legal   46       Singapore  141721
0         1011      Satender Kumar  Data Analyst   24  London, Canada   75719
1         1012              data 1            HR   40     Los Angeles   96413
2         1013         Chris Evans            IT   37        New York   74042
3         1014     Natalie Portman     Marketing   49         Chicago  108467
4         1015         Tom Holland       Finance   27         Houston   65822
5         1016         Emma Watson         Sales   56         Phoenix  114854
```

6	1017	Daniel Radcliffe	R&D	33	Philadelphia	117933	
7	1018	Scarlett Johansson	Operations	41	San Antonio	87342	
8	1019	Robert Downey Jr.	Legal	51	San Diego	72775	
9	1020	Mark Ruffalo	Support	42	Dallas	142745	

	Years with Company	Position	Performance Score	Bonus	\
0	10	Data Analyst	4	6419	
1	5	Developer	4	4746	
2	1	Analyst	1	1447	
3	9	Designer	3	7196	
4	13	Consultant	3	2218	
5	6	Engineer	2	7973	
6	13	Scientist	4	7565	
7	9	Support Agent	3	1800	
8	7	Admin Assistant	4	1189	
9	2	Lawyer	2	1654	
0	12	Data Analyst	2	3187	
1	12	HR Manager	1	10993	
2	14	IT Specialist	4	10834	
3	11	Marketing Coordinator	4	2760	
4	12	Financial Analyst	2	6263	
5	5	Sales Manager	3	10287	
6	17	Research Scientist	3	8290	
7	8	Operations Manager	3	10047	
8	6	Legal Advisor	2	6510	
9	10	Support Specialist	1	5533	

	Gender	Marital Status	Education	Hire Date	Overtime Hours	\
0	Male	Single	Bachelor	2019-06-12	10	
1	Male	Single	Master	2015-07-23	15	
2	Female	Married	PhD	2012-09-05	0	
3	Male	Single	Bachelor	2018-11-30	18	
4	Female	Single	Master	2013-05-19	1	
5	Male	Married	PhD	2019-02-14	0	
6	Female	Married	Bachelor	2020-08-21	3	
7	Male	Single	Master	2016-06-03	6	
8	Female	Married	Bachelor	2014-01-28	15	
9	Male	Single	Master	2017-03-15	3	
0	Male	Single	Master	2018-07-15	17	
1	Male	Married	Bachelor	2014-03-22	8	
2	Female	Single	Master	2011-10-12	12	
3	Female	Single	PhD	2017-04-17	10	
4	Male	Married	Bachelor	2015-09-23	13	
5	Female	Single	Master	2016-11-01	15	
6	Male	Single	PhD	2019-05-11	8	
7	Female	Married	Bachelor	2020-07-08	15	
8	Male	Single	Master	2013-08-19	7	
9	Male	Married	PhD	2012-01-09	2	

	Sick Days Taken	Vacation Days Taken	Training Hours	Certifications	\
0	2	19	15	Yes	
1	7	14	31	No	
2	5	7	37	Yes	
3	3	11	12	No	
4	7	17	30	Yes	
5	1	16	24	No	
6	0	7	49	Yes	
7	0	6	10	Yes	
8	1	11	38	No	
9	7	19	40	Yes	
0	5	12	21	Yes	
1	3	18	41	Yes	
2	4	8	22	No	
3	7	20	42	Yes	
4	6	11	35	No	
5	0	20	35	Yes	
6	1	15	48	No	
7	7	9	50	Yes	
8	7	20	35	Yes	
9	4	17	15	No	

	Supervisor
0	Anna Smith
1	Brian Adams
2	Clara Jones
3	Daniel Martin
4	Eva Rodriguez
5	Frank Bell
6	Grace Moore
7	Hannah Lewis
8	Ivan Scott
9	Jake Miller
0	John Smith

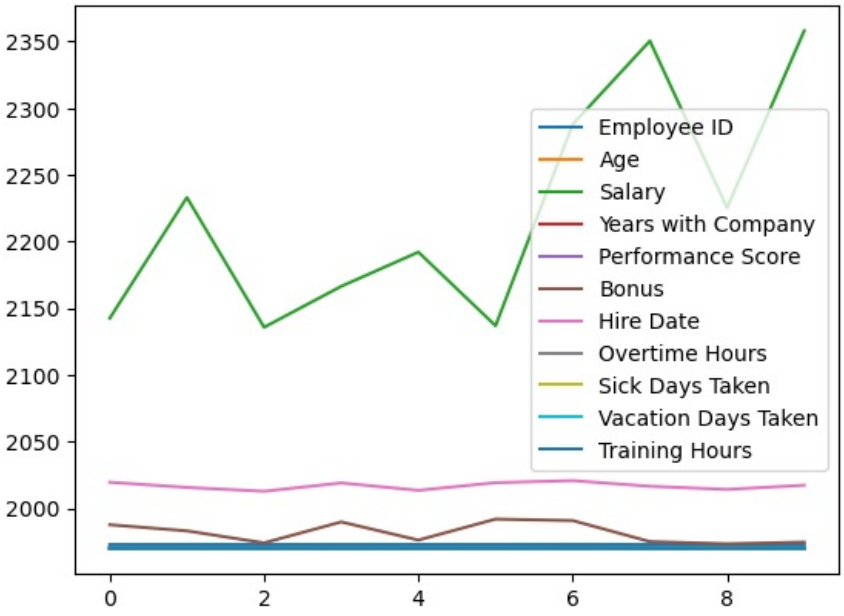
```
1    Michael Johnson
2  Patricia Williams
3        Linda Brown
4      Barbara Jones
5   Elizabeth Garcia
6     Susan Martinez
7  Jessica Hernandez
8        Sarah Lopez
9       Karen Wilson
```

In [52]:
```
df.info()
```

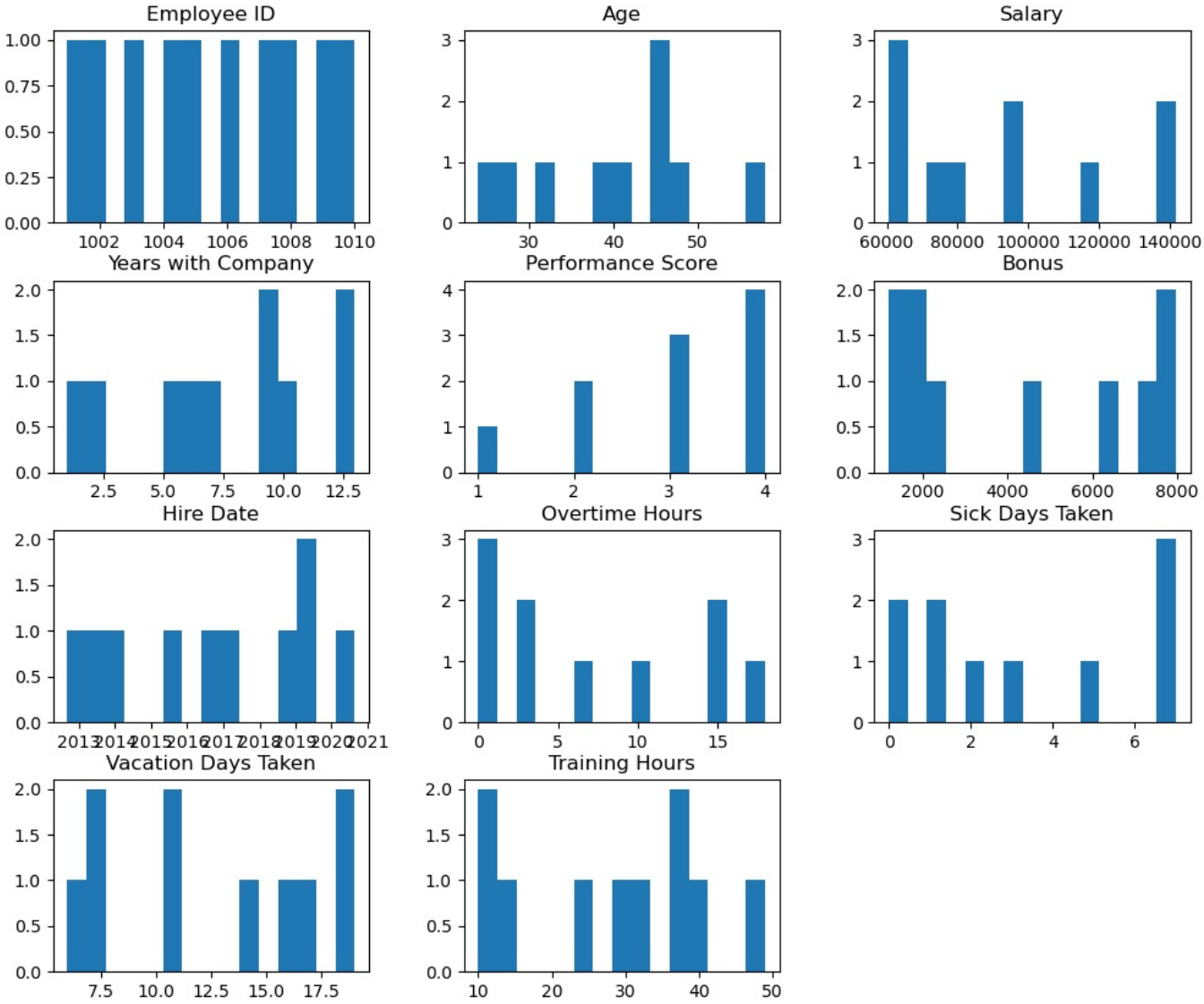
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 20 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Employee ID          10 non-null     int32
 1   Employee Name        10 non-null     object
 2   Department           10 non-null     object
 3   Age                  10 non-null     int64
 4   Location             10 non-null     object
 5   Salary               10 non-null     int32
 6   Years with Company   10 non-null     int32
 7   Position             10 non-null     object
 8   Performance Score    10 non-null     int32
 9   Bonus                10 non-null     int32
 10  Gender               10 non-null     object
 11  Marital Status       10 non-null     object
 12  Education            10 non-null     object
 13  Hire Date            10 non-null     datetime64[ns]
 14  Overtime Hours       10 non-null     int32
 15  Sick Days Taken      10 non-null     int32
 16  Vacation Days Taken  10 non-null     int32
 17  Training Hours       10 non-null     int32
 18  Certifications       10 non-null     object
 19  Supervisor           10 non-null     object
dtypes: datetime64[ns](1), int32(9), int64(1), object(9)
memory usage: 1.3+ KB
```

In [55]:
```
df.plot()
```

Out[55]: <Axes: >



In [57]:
```
df.hist(figsize=(12, 10), bins=15, grid=False)

# Display the plots
plt.show()
```



In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Panda|Compare DataFrame

In [1]:
```
import pandas as pd  # For data manipulation and analysis
import numpy as np  # For numerical computations
import matplotlib.pyplot as plt  # For creating static, animated, and interactive visualizations
import seaborn as sns  # For statistical data visualization based on Matplotlib
import scipy  # For scientific and technical computing (including optimization, integration, and statistics)
```

In [3]:
```
import pandas as pd
import numpy as np
# Creating realistic data for employees
data = {
    'Employee ID': np.arange(1001, 1011),
    'Employee Name': ['Satender Kumar', 'data 1', 'Jane Smith', 'Robert Brown', 'Emily Davis', 'Michael Wilson'
    'Department': ['Data Analyst', 'IT', 'Finance', 'Marketing', 'Sales', 'Operations', 'R&D', 'Support', 'Admi
    'Age': [24, np.random.randint(25, 60), np.random.randint(25, 60), np.random.randint(25, 60), np.random.rand
    'Location': ['London, Canada', 'Toronto', 'London', 'Sydney', 'San Francisco', 'Paris', 'Berlin', 'Tokyo',
    'Salary': np.random.randint(50000, 150000, size=10),
    'Years with Company': np.random.randint(1, 15, size=10),
    'Position': ['Data Analyst', 'Developer', 'Analyst', 'Designer', 'Consultant', 'Engineer', 'Scientist', 'Su
    'Performance Score': np.random.randint(1, 5, size=10),
    'Bonus': np.random.randint(1000, 10000, size=10),
    'Gender': ['Male', 'Male', 'Female', 'Male', 'Female', 'Male', 'Female', 'Male', 'Female', 'Male'],
    'Marital Status': ['Single', 'Single', 'Married', 'Single', 'Single', 'Married', 'Married', 'Single', 'Marr
    'Education': ['Bachelor', 'Master', 'PhD', 'Bachelor', 'Master', 'PhD', 'Bachelor', 'Master', 'Bachelor', '
    'Hire Date': pd.to_datetime(['2019-06-12', '2015-07-23', '2012-09-05', '2018-11-30', '2013-05-19', '2019-02
    'Overtime Hours': np.random.randint(0, 20, size=10),
    'Sick Days Taken': np.random.randint(0, 10, size=10),
    'Vacation Days Taken': np.random.randint(5, 20, size=10),
    'Training Hours': np.random.randint(10, 50, size=10),
    'Certifications': ['Yes', 'No', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'No', 'Yes'],
    'Supervisor': ['Anna Smith', 'Brian Adams', 'Clara Jones', 'Daniel Martin', 'Eva Rodriguez', 'Frank Bell',
}

# Creating the DataFrame
df = pd.DataFrame(data)
```

In [5]:
```
df
```

Out[5]:

	Employee ID	Employee Name	Department	Age	Location	Salary	Years with Company	Position	Performance Score	Bonus	Gender	Marital Status	Educat
0	1001	Satender Kumar	Data Analyst	24	London, Canada	113479	3	Data Analyst	4	9480	Male	Single	Bach
1	1002	data 1	IT	34	Toronto	127453	9	Developer	4	6577	Male	Single	Ma
2	1003	Jane Smith	Finance	27	London	118316	7	Analyst	1	4972	Female	Married	F
3	1004	Robert Brown	Marketing	59	Sydney	109725	10	Designer	3	4624	Male	Single	Bach
4	1005	Emily Davis	Sales	36	San Francisco	99006	14	Consultant	3	3218	Female	Single	Ma
5	1006	Michael Wilson	Operations	36	Paris	123386	8	Engineer	1	4450	Male	Married	F
6	1007	Sarah Taylor	R&D	54	Berlin	77293	13	Scientist	2	2823	Female	Married	Bach
7	1008	David Lee	Support	32	Tokyo	65542	9	Support Agent	2	1486	Male	Single	Ma
8	1009	Laura Johnson	Admin	50	Dubai	87223	9	Admin Assistant	2	1705	Female	Married	Bach
9	1010	James White	Legal	25	Singapore	110051	5	Lawyer	2	8724	Male	Single	Ma

In [6]:
```
# Creating realistic data for a second set of employees
data1 = {
    'Employee ID': np.arange(1011, 1021),
    'Employee Name': ['Satender Kumar', 'data 1', 'Chris Evans', 'Natalie Portman', 'Tom Holland', 'Emma Watson
    'Department': ['Data Analyst', 'HR', 'IT', 'Marketing', 'Finance', 'Sales', 'R&D', 'Operations', 'Legal', '
    'Age': [24, np.random.randint(25, 60), np.random.randint(25, 60), np.random.randint(25, 60), np.random.rand
    'Location': ['London, Canada', 'Los Angeles', 'New York', 'Chicago', 'Houston', 'Phoenix', 'Philadelphia',
    'Salary': np.random.randint(60000, 160000, size=10),
    'Years with Company': np.random.randint(1, 20, size=10),
```

```
    'Position': ['Data Analyst', 'HR Manager', 'IT Specialist', 'Marketing Coordinator', 'Financial Analyst', '
    'Performance Score': np.random.randint(1, 5, size=10),
    'Bonus': np.random.randint(2000, 12000, size=10),
    'Gender': ['Male', 'Male', 'Female', 'Female', 'Male', 'Female', 'Male', 'Female', 'Male', 'Male'],
    'Marital Status': ['Single', 'Married', 'Single', 'Single', 'Married', 'Single', 'Single', 'Married', 'Sing
    'Education': ['Master', 'Bachelor', 'Master', 'PhD', 'Bachelor', 'Master', 'PhD', 'Bachelor', 'Master', 'Phl
    'Hire Date': pd.to_datetime(['2018-07-15', '2014-03-22', '2011-10-12', '2017-04-17', '2015-09-23', '2016-11
    'Overtime Hours': np.random.randint(0, 25, size=10),
    'Sick Days Taken': np.random.randint(0, 8, size=10),
    'Vacation Days Taken': np.random.randint(7, 22, size=10),
    'Training Hours': np.random.randint(15, 55, size=10),
    'Certifications': ['Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'No'],
    'Supervisor': ['John Smith', 'Michael Johnson', 'Patricia Williams', 'Linda Brown', 'Barbara Jones', 'Eliza
}

# Creating the second DataFrame
df1 = pd.DataFrame(data1)
```

In [8]:
```
df1
```

Out[8]:

	Employee ID	Employee Name	Department	Age	Location	Salary	Years with Company	Position	Performance Score	Bonus	Gender	Marital Status	Edu
0	1011	Satender Kumar	Data Analyst	24	London, Canada	125763	9	Data Analyst	4	9043	Male	Single	
1	1012	data 1	HR	29	Los Angeles	149163	18	HR Manager	4	5174	Male	Married	Ba
2	1013	Chris Evans	IT	58	New York	149334	15	IT Specialist	4	4389	Female	Single	
3	1014	Natalie Portman	Marketing	41	Chicago	152579	13	Marketing Coordinator	4	3100	Female	Single	
4	1015	Tom Holland	Finance	55	Houston	87396	11	Financial Analyst	1	5710	Male	Married	Ba
5	1016	Emma Watson	Sales	34	Phoenix	132586	8	Sales Manager	1	2777	Female	Single	
6	1017	Daniel Radcliffe	R&D	56	Philadelphia	143832	10	Research Scientist	2	6264	Male	Single	
7	1018	Scarlett Johansson	Operations	35	San Antonio	61204	1	Operations Manager	4	9008	Female	Married	Ba
8	1019	Robert Downey Jr.	Legal	26	San Diego	69724	17	Legal Advisor	4	7002	Male	Single	
9	1020	Mark Ruffalo	Support	33	Dallas	134630	11	Support Specialist	1	2651	Male	Married	

In [11]:
```
df.head()
```

Out[11]:

	Employee ID	Employee Name	Department	Age	Location	Salary	Years with Company	Position	Performance Score	Bonus	Gender	Marital Status	Educat
0	1001	Satender Kumar	Data Analyst	24	London, Canada	113479	3	Data Analyst	4	9480	Male	Single	Bache
1	1002	data 1	IT	34	Toronto	127453	9	Developer	4	6577	Male	Single	Mas
2	1003	Jane Smith	Finance	27	London	118316	7	Analyst	1	4972	Female	Married	F
3	1004	Robert Brown	Marketing	59	Sydney	109725	10	Designer	3	4624	Male	Single	Bache
4	1005	Emily Davis	Sales	36	San Francisco	99006	14	Consultant	3	3218	Female	Single	Mas

In [13]:
```
df1.head()
```

Out[13]:

	Employee ID	Employee Name	Department	Age	Location	Salary	Years with Company	Position	Performance Score	Bonus	Gender	Marital Status	Educa
0	1011	Satender Kumar	Data Analyst	24	London, Canada	125763	9	Data Analyst	4	9043	Male	Single	Ma
1	1012	data 1	HR	29	Los Angeles	149163	18	HR Manager	4	5174	Male	Married	Bach
2	1013	Chris Evans	IT	58	New York	149334	15	IT Specialist	4	4389	Female	Single	Ma
3	1014	Natalie Portman	Marketing	41	Chicago	152579	13	Marketing Coordinator	4	3100	Female	Single	
4	1015	Tom Holland	Finance	55	Houston	87396	11	Financial Analyst	1	5710	Male	Married	Bach

In [15]:
```
# Check if df and df1 are exactly the same
are_identical = df.equals(df1)
print(f"Are df and df1 identical? {are_identical}")
```

```
Are df and df1 identical? False
```

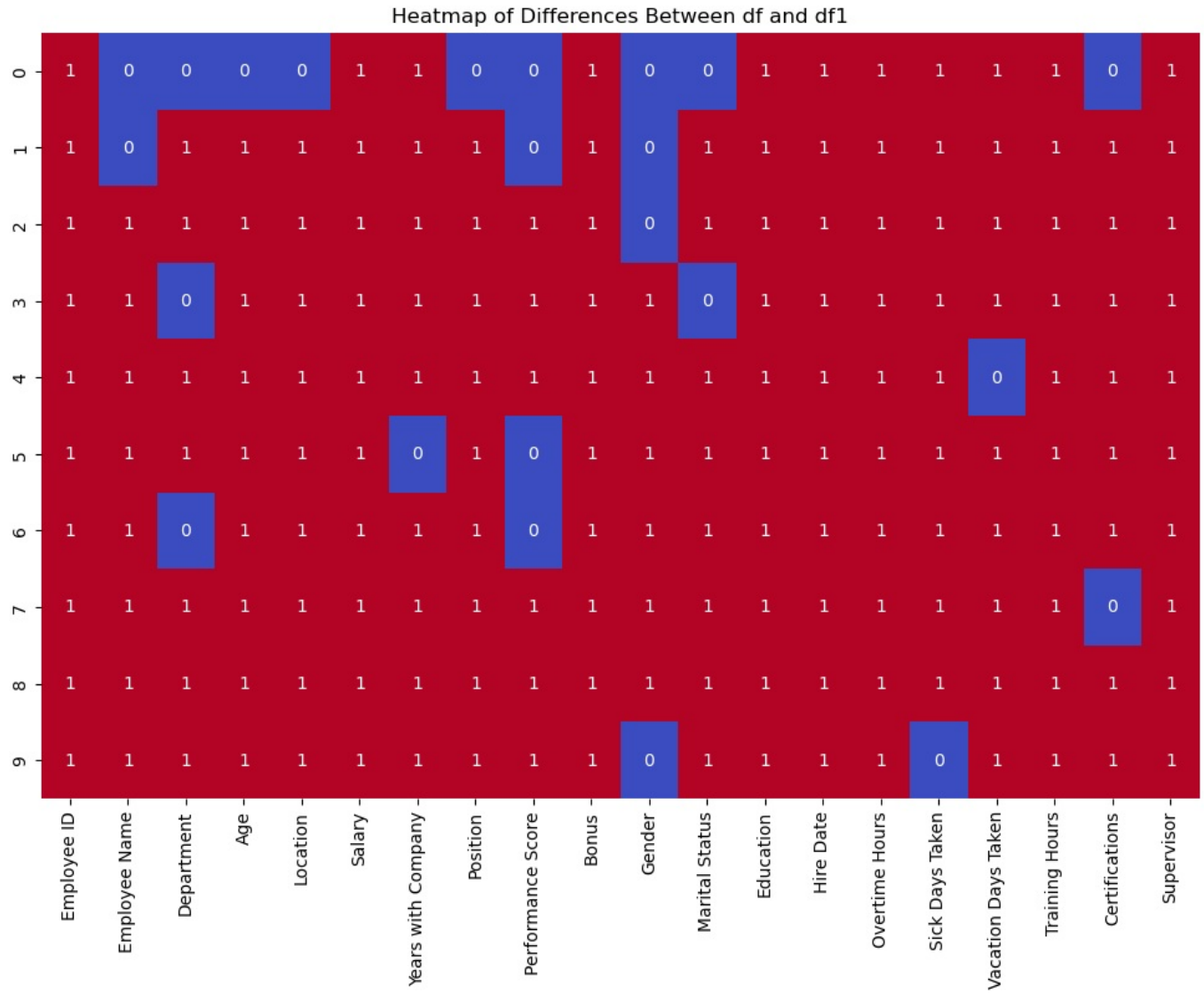
In [17]:
```
# Find differences between df and df1
comparison_df = df.compare(df1, keep_shape=True, keep_equal=True)
print("Differences between df and df1:")
print(comparison_df)
```

```
Differences between df and df1:
  Employee ID         Employee Name                       Department  \
         self other            self                other          self
0        1001  1011  Satender Kumar      Satender Kumar  Data Analyst
1        1002  1012          data 1              data 1            IT
2        1003  1013      Jane Smith         Chris Evans       Finance
3        1004  1014    Robert Brown     Natalie Portman     Marketing
4        1005  1015     Emily Davis         Tom Holland         Sales
5        1006  1016  Michael Wilson         Emma Watson    Operations
6        1007  1017    Sarah Taylor    Daniel Radcliffe           R&D
7        1008  1018       David Lee  Scarlett Johansson       Support
8        1009  1019   Laura Johnson   Robert Downey Jr.         Admin
9        1010  1020     James White        Mark Ruffalo         Legal

                Age             Location                        ...  \
           other self other          self           other       ...
0  Data Analyst   24    24  London, Canada  London, Canada     ...
1            HR   34    29         Toronto     Los Angeles     ...
2            IT   27    58          London        New York     ...
3     Marketing   59    41          Sydney         Chicago     ...
4       Finance   36    55   San Francisco         Houston     ...
5         Sales   36    34           Paris         Phoenix     ...
6           R&D   54    56          Berlin    Philadelphia     ...
7    Operations   32    35           Tokyo     San Antonio     ...
8         Legal   50    26           Dubai       San Diego     ...
9       Support   25    33       Singapore          Dallas     ...

  Sick Days Taken       Vacation Days Taken       Training Hours        \
             self other                self other           self other
0               9     7                12    15            19    53
1               2     5                15    19            31    17
2               2     6                10    13            30    45
3               8     4                14    20            29    34
4               9     1                18    18            14    17
5               3     7                 7    14            14    48
6               5     4                 6     8            11    46
7               4     0                15    16            21    54
8               8     4                15     7            44    18
9               6     6                 6    21            45    31

  Certifications           Supervisor
            self other           self             other
0            Yes   Yes     Anna Smith         John Smith
1             No   Yes    Brian Adams    Michael Johnson
2            Yes    No    Clara Jones  Patricia Williams
3             No   Yes  Daniel Martin        Linda Brown
4            Yes    No  Eva Rodriguez      Barbara Jones
5             No   Yes     Frank Bell   Elizabeth Garcia
6            Yes    No    Grace Moore     Susan Martinez
7            Yes   Yes   Hannah Lewis  Jessica Hernandez
8             No   Yes     Ivan Scott        Sarah Lopez
9            Yes    No    Jake Miller       Karen Wilson

[10 rows x 40 columns]
```

In [20]:
```
# Identify rows that differ between df and df1
differing_rows = df[df.ne(df1).any(axis=1)]
print("Rows that differ between df and df1:")
print(differing_rows)
```

```
Rows that differ between df and df1:
   Employee ID   Employee Name    Department  Age        Location  Salary  \
0         1001  Satender Kumar  Data Analyst   24  London, Canada  113479
1         1002          data 1            IT   34         Toronto  127453
2         1003      Jane Smith       Finance   27          London  118316
3         1004    Robert Brown     Marketing   59          Sydney  109725
4         1005     Emily Davis         Sales   36   San Francisco   99006
5         1006  Michael Wilson    Operations   36           Paris  123386
6         1007    Sarah Taylor           R&D   54          Berlin   77293
7         1008       David Lee       Support   32           Tokyo   65542
8         1009   Laura Johnson         Admin   50           Dubai   87223
9         1010     James White         Legal   25       Singapore  110051

   Years with Company         Position  Performance Score  Bonus  Gender  \
0                   3     Data Analyst                  4   9480    Male
1                   9        Developer                  4   6577    Male
2                   7          Analyst                  1   4972  Female
3                  10         Designer                  3   4624    Male
4                  14       Consultant                  3   3218  Female
5                   8         Engineer                  1   4450    Male
6                  13        Scientist                  2   2823  Female
7                   9    Support Agent                  2   1486    Male
8                   9  Admin Assistant                  2   1705  Female
9                   5           Lawyer                  2   8724    Male

  Marital Status Education  Hire Date  Overtime Hours  Sick Days Taken  \
0         Single  Bachelor 2019-06-12              18                9
1         Single    Master 2015-07-23              11                2
2        Married       PhD 2012-09-05               6                2
3         Single  Bachelor 2018-11-30              17                8
4         Single    Master 2013-05-19              16                9
5        Married       PhD 2019-02-14              14                3
6        Married  Bachelor 2020-08-21               4                5
7         Single    Master 2016-06-03               3                4
8        Married  Bachelor 2014-01-28              13                8
9         Single    Master 2017-03-15               6                6

   Vacation Days Taken  Training Hours Certifications     Supervisor
0                   12              19            Yes     Anna Smith
1                   15              31             No    Brian Adams
2                   10              30            Yes    Clara Jones
3                   14              29             No  Daniel Martin
4                   18              14            Yes  Eva Rodriguez
5                    7              14             No     Frank Bell
6                    6              11            Yes    Grace Moore
7                   15              21            Yes   Hannah Lewis
8                   15              44             No     Ivan Scott
9                    6              45            Yes    Jake Miller
```

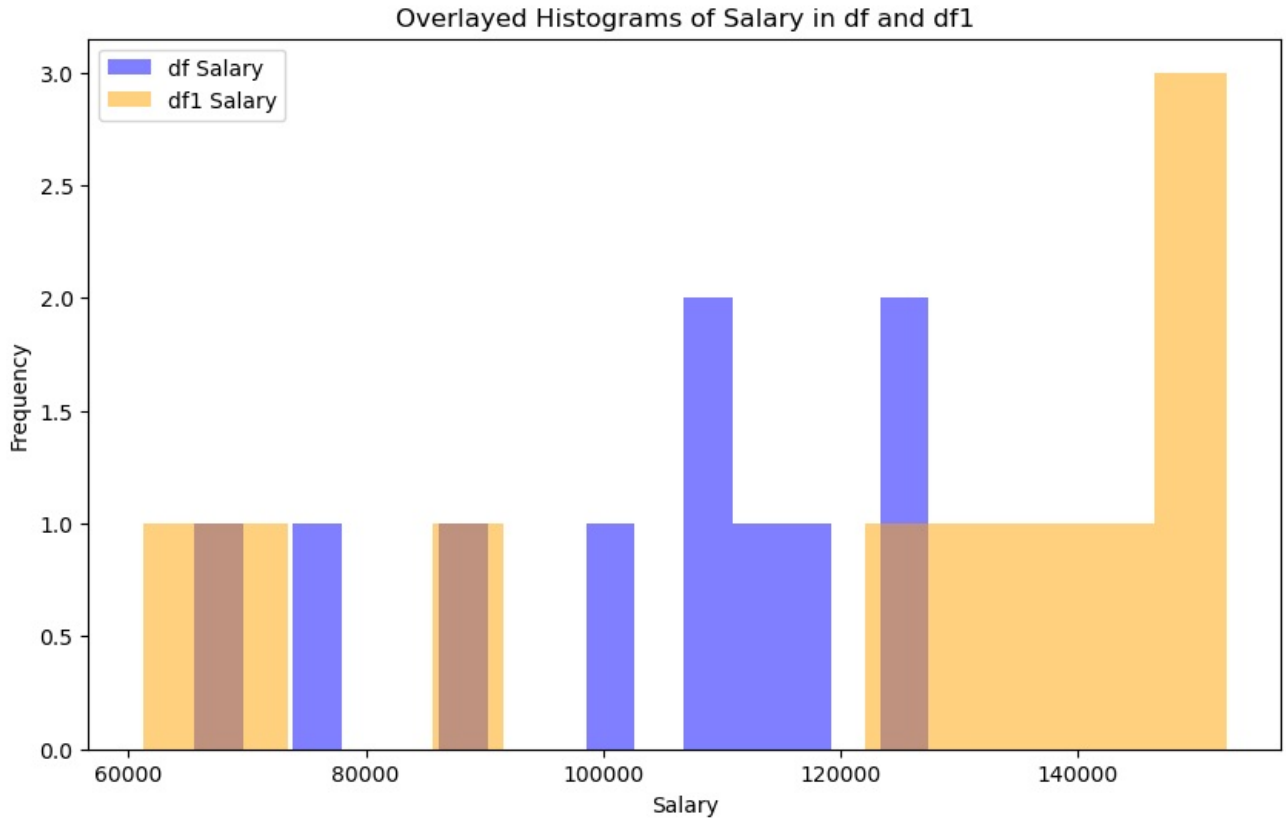
In [22]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Create a heatmap to visualize differences between df and df1
diff = df.ne(df1).astype(int)  # 1 where different, 0 where the same
plt.figure(figsize=(12, 8))
sns.heatmap(diff, cmap='coolwarm', cbar=False, annot=True)
plt.title('Heatmap of Differences Between df and df1')
plt.show()
```



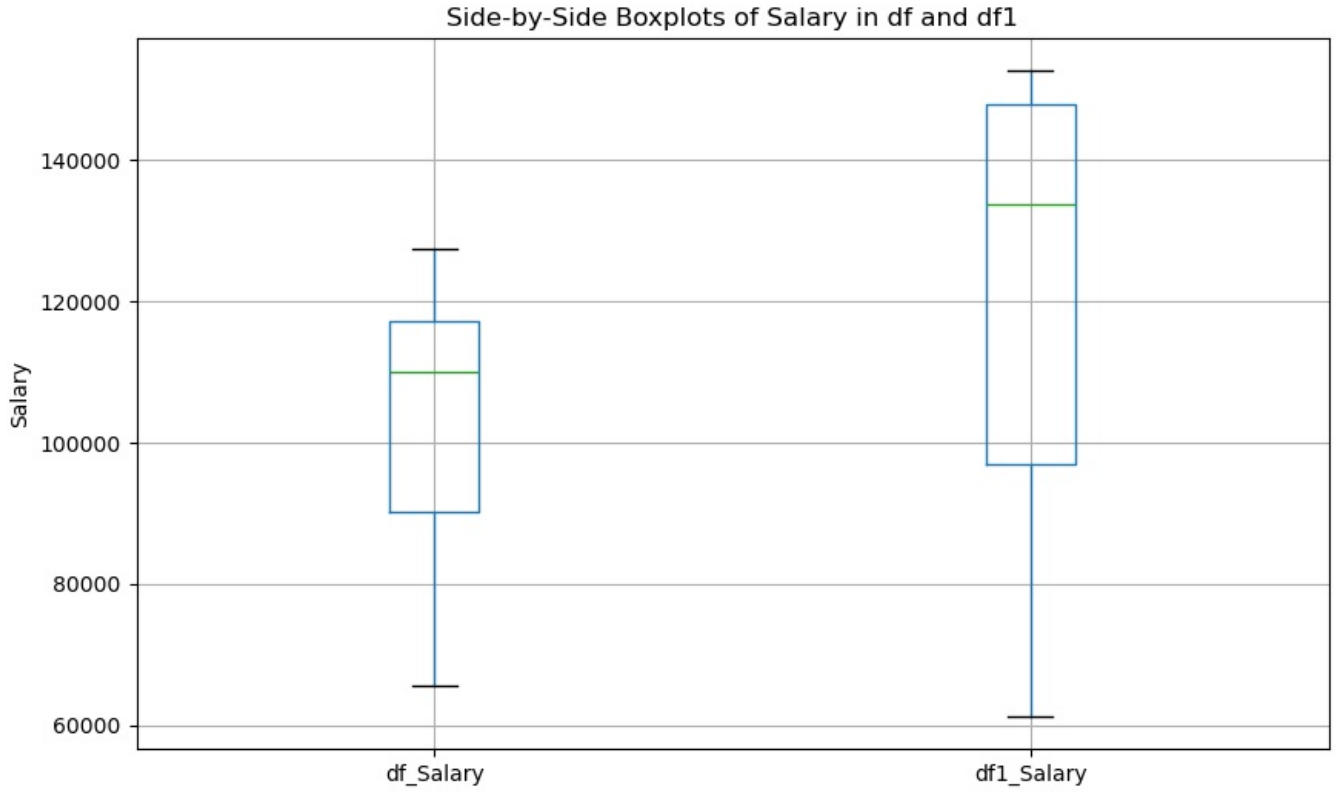
In [24]:
```
# Summarize the number of differences by column
summary = df.ne(df1).sum()
print("Summary of differences by column:")
print(summary)
```

```
Summary of differences by column:
Employee ID            10
Employee Name           8
Department              7
Age                     9
Location                9
Salary                 10
Years with Company      9
Position                9
Performance Score       6
Bonus                  10
Gender                  6
Marital Status          8
Education              10
Hire Date              10
Overtime Hours         10
Sick Days Taken         9
Vacation Days Taken     9
Training Hours         10
Certifications          8
Supervisor             10
dtype: int64
```

In [26]:
```
import matplotlib.pyplot as plt

# Overlayed histograms for 'Salary' in df and df1
plt.figure(figsize=(10, 6))
plt.hist(df['Salary'], bins=15, alpha=0.5, label='df Salary', color='blue')
plt.hist(df1['Salary'], bins=15, alpha=0.5, label='df1 Salary', color='orange')
plt.title('Overlayed Histograms of Salary in df and df1')
plt.xlabel('Salary')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```

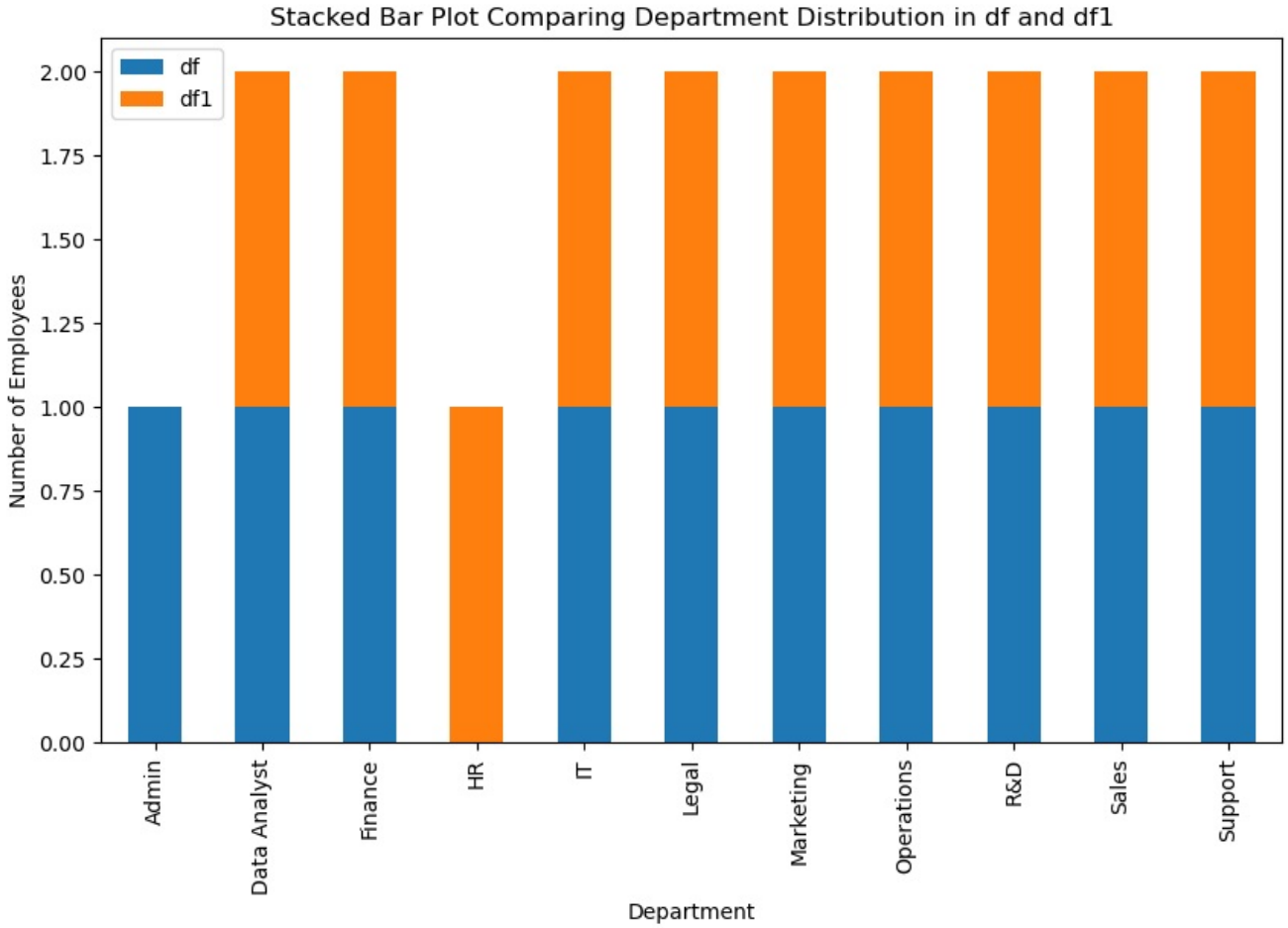


In [29]:
```python
# Combine the data into a single DataFrame for side-by-side boxplots
combined = pd.DataFrame({
    'df_Salary': df['Salary'],
    'df1_Salary': df1['Salary']
})

# Plot the boxplots
plt.figure(figsize=(10, 6))
combined.boxplot()
plt.title('Side-by-Side Boxplots of Salary in df and df1')
plt.ylabel('Salary')
plt.show()
```

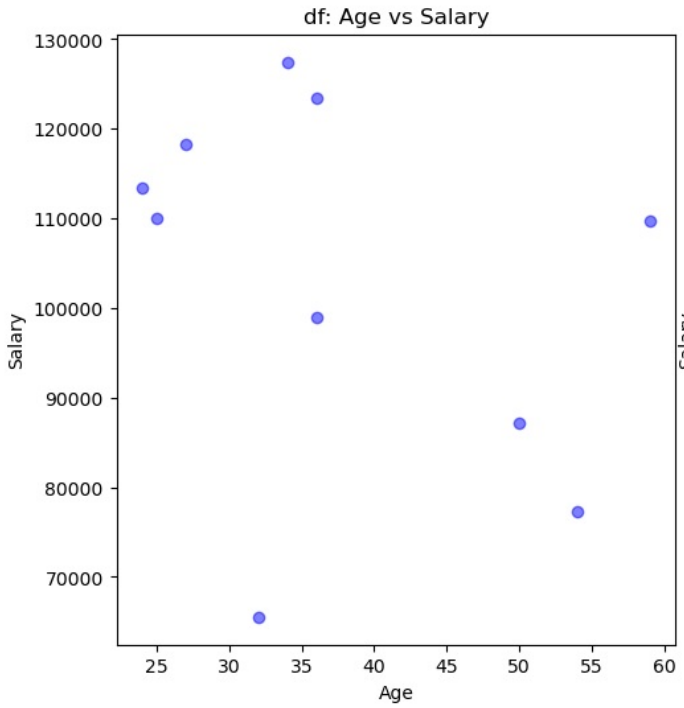


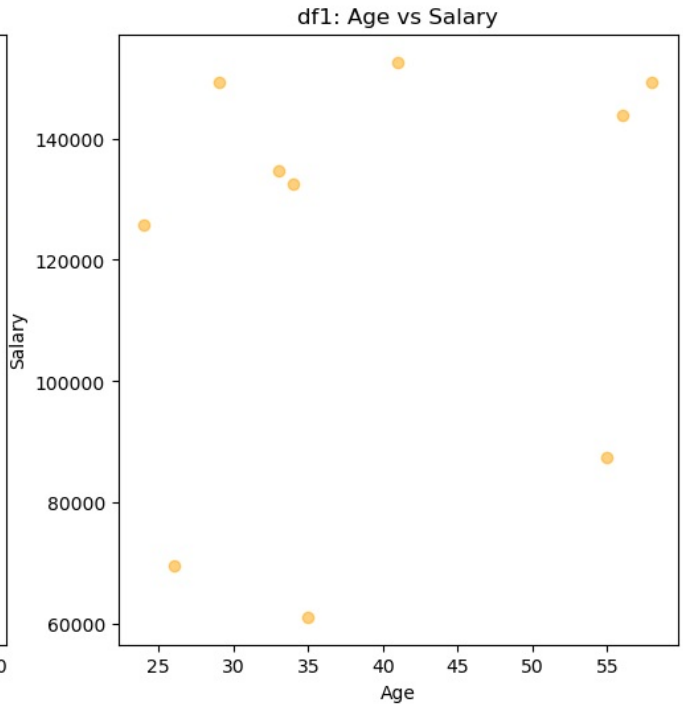
In [31]:
```python
# Group by Department and count in both DataFrames
df_dept_counts = df['Department'].value_counts()
df1_dept_counts = df1['Department'].value_counts()

# Combine into a single DataFrame
```

```
dept_comparison = pd.DataFrame({'df': df_dept_counts, 'df1': df1_dept_counts})

# Plot stacked bar plot
dept_comparison.plot(kind='bar', stacked=True, figsize=(10, 6))
plt.title('Stacked Bar Plot Comparing Department Distribution in df and df1')
plt.ylabel('Number of Employees')
plt.show()
```

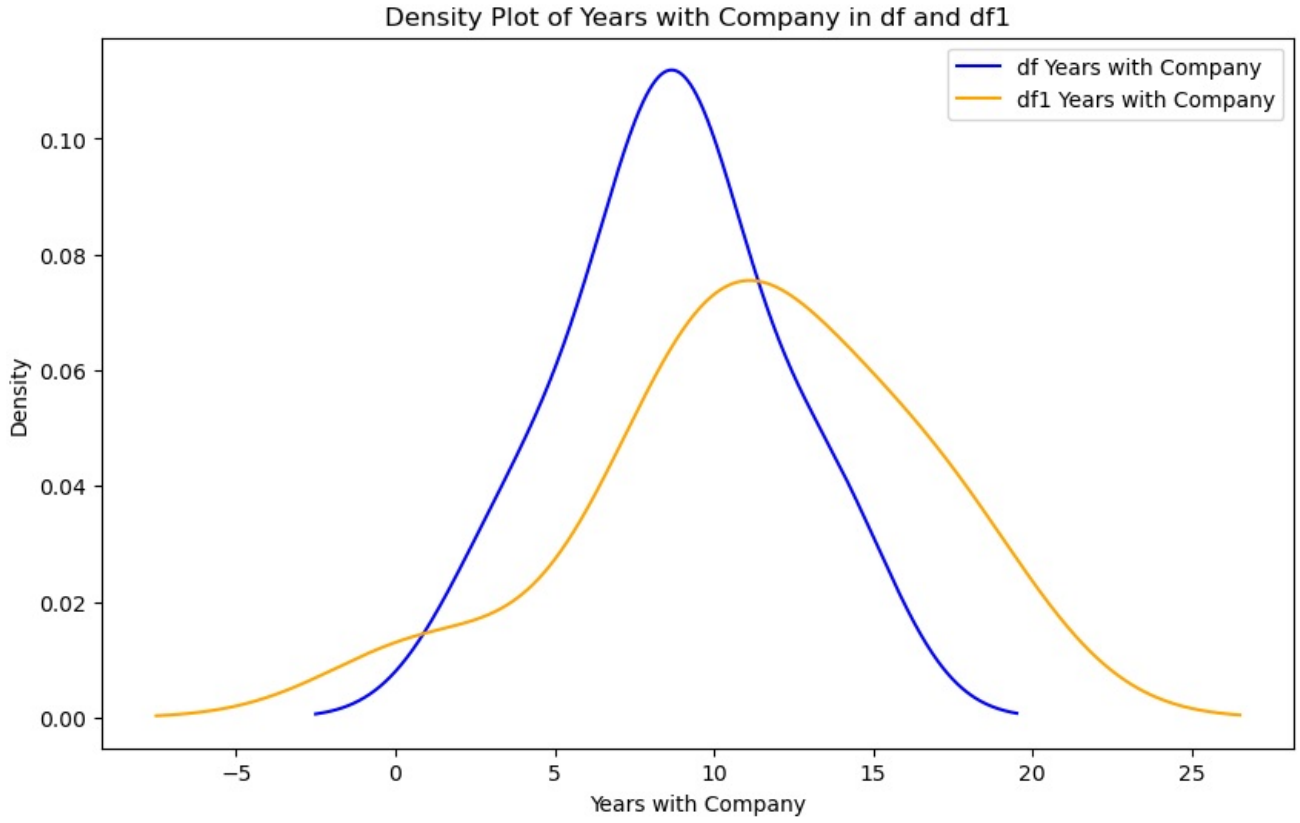


In [33]:
```
# Scatter plot comparison for 'Age' vs 'Salary' in both DataFrames
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.scatter(df['Age'], df['Salary'], color='blue', alpha=0.5)
plt.title('df: Age vs Salary')
plt.xlabel('Age')
plt.ylabel('Salary')

plt.subplot(1, 2, 2)
plt.scatter(df1['Age'], df1['Salary'], color='orange', alpha=0.5)
plt.title('df1: Age vs Salary')
plt.xlabel('Age')
plt.ylabel('Salary')

plt.show()
```

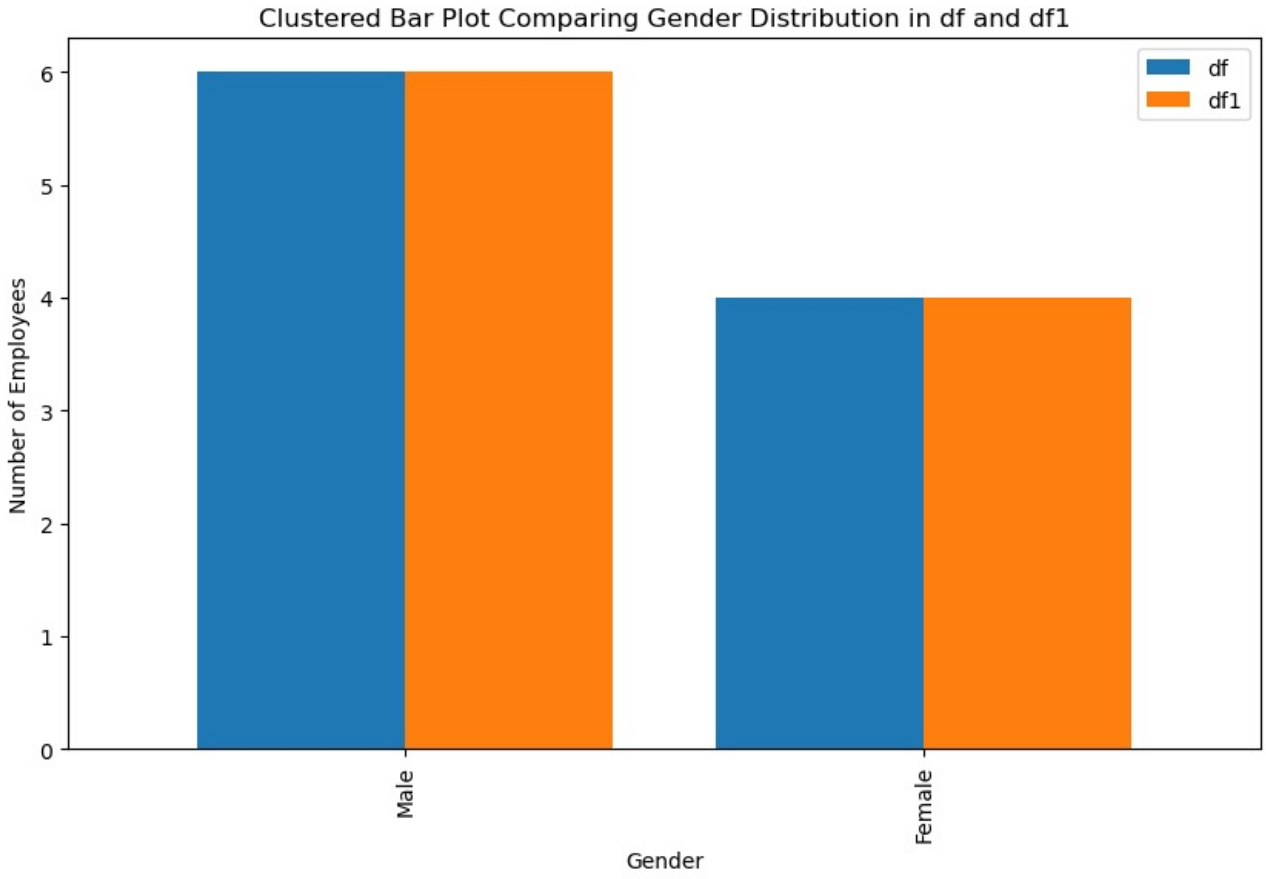




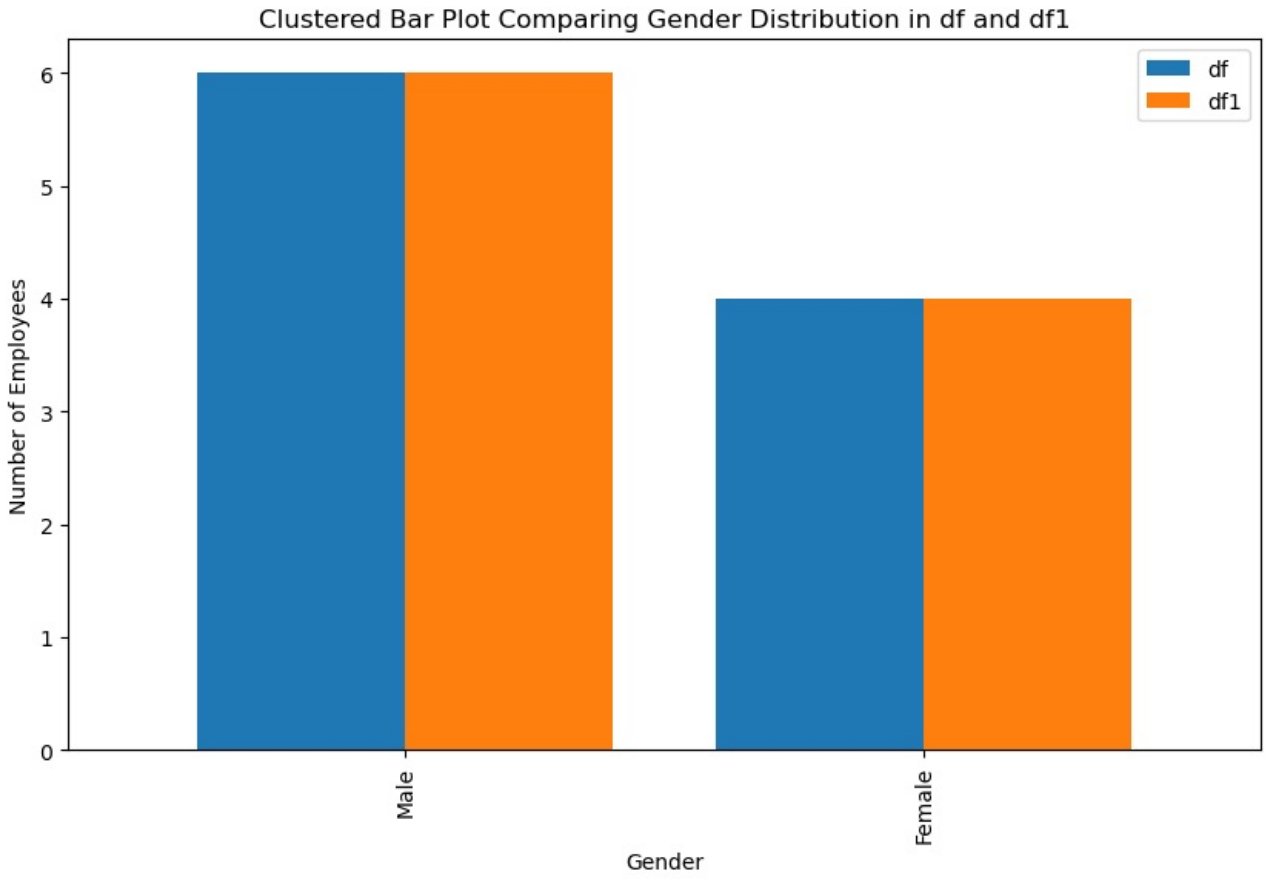
In [35]:
```
# Density plot for 'Years with Company' in df and df1
plt.figure(figsize=(10, 6))
df['Years with Company'].plot(kind='density', label='df Years with Company', color='blue')
df1['Years with Company'].plot(kind='density', label='df1 Years with Company', color='orange')
plt.title('Density Plot of Years with Company in df and df1')
plt.xlabel('Years with Company')
plt.legend()
plt.show()
```



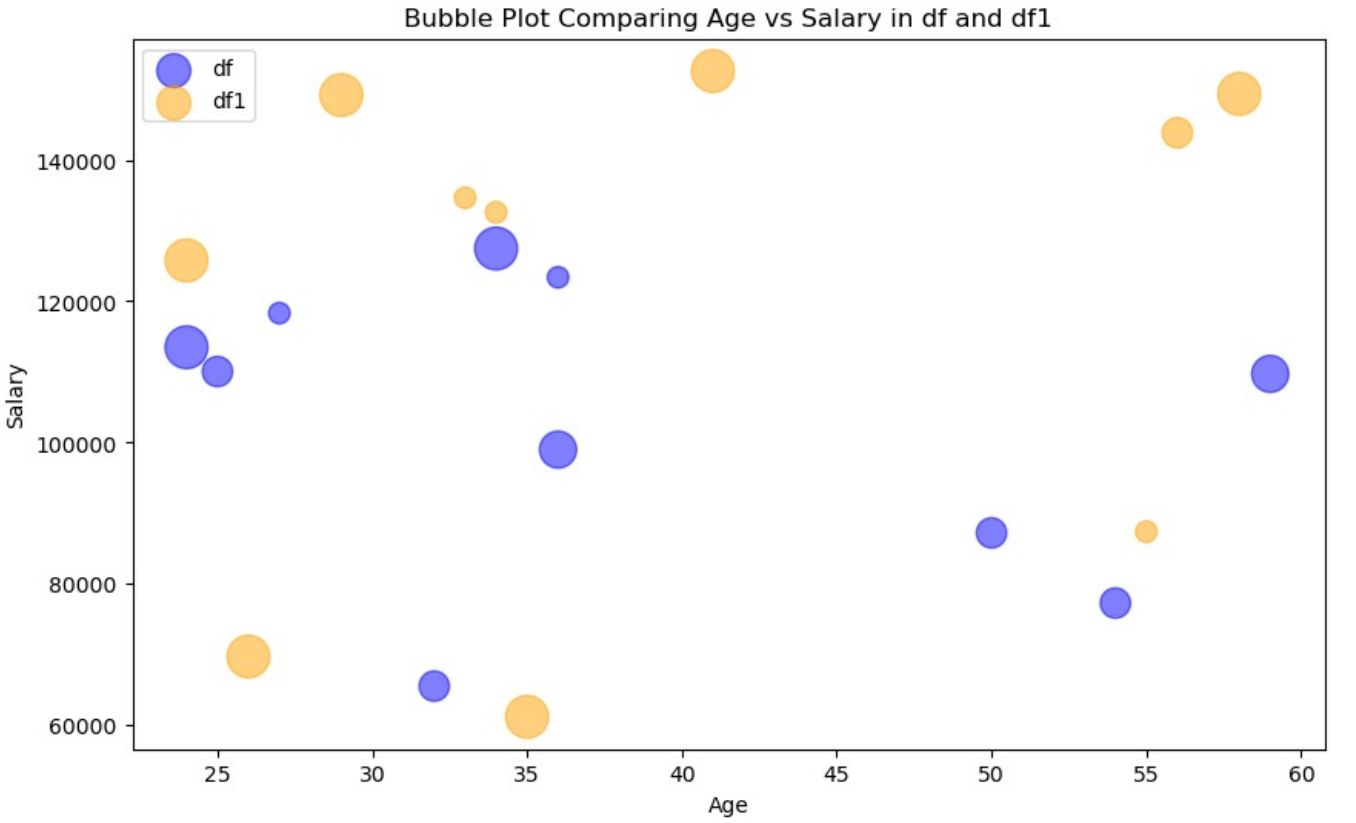
In [37]:
```
# Group by Gender and count in both DataFrames
df_gender_counts = df['Gender'].value_counts()
df1_gender_counts = df1['Gender'].value_counts()

# Combine into a single DataFrame
gender_comparison = pd.DataFrame({'df': df_gender_counts, 'df1': df1_gender_counts})

# Plot clustered bar plot
gender_comparison.plot(kind='bar', width=0.8, figsize=(10, 6))
plt.title('Clustered Bar Plot Comparing Gender Distribution in df and df1')
plt.ylabel('Number of Employees')
plt.show()
```



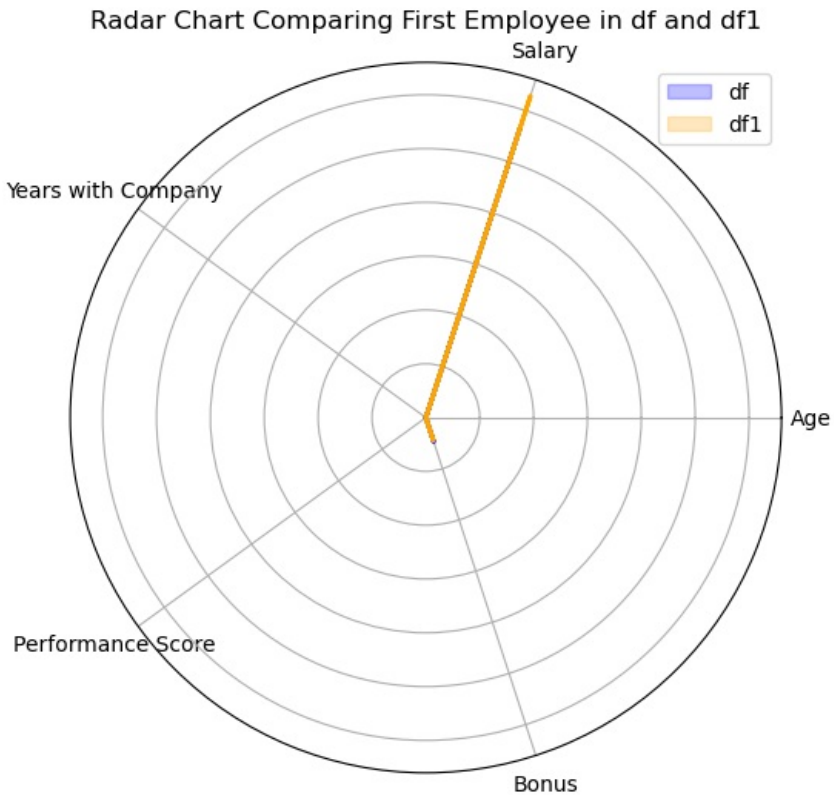
In [39]:
```
# Group by Gender and count in both DataFrames
df_gender_counts = df['Gender'].value_counts()
df1_gender_counts = df1['Gender'].value_counts()

# Combine into a single DataFrame
gender_comparison = pd.DataFrame({'df': df_gender_counts, 'df1': df1_gender_counts})

# Plot clustered bar plot
gender_comparison.plot(kind='bar', width=0.8, figsize=(10, 6))
plt.title('Clustered Bar Plot Comparing Gender Distribution in df and df1')
plt.ylabel('Number of Employees')
plt.show()
```



In [41]:
```
# Bubble plot for Age vs Salary with Performance Score as bubble size
plt.figure(figsize=(10, 6))

plt.scatter(df['Age'], df['Salary'], s=df['Performance Score']*100, alpha=0.5, label='df', color='blue')
plt.scatter(df1['Age'], df1['Salary'], s=df1['Performance Score']*100, alpha=0.5, label='df1', color='orange')

plt.title('Bubble Plot Comparing Age vs Salary in df and df1')
plt.xlabel('Age')
plt.ylabel('Salary')
plt.legend()
plt.show()
```



In [43]:
```
# Comparing first employee in both DataFrames using a radar chart
categories = ['Age', 'Salary', 'Years with Company', 'Performance Score', 'Bonus']
df_values = df.loc[0, categories].values.flatten().tolist()
df1_values = df1.loc[0, categories].values.flatten().tolist()

# Closing the radar chart by adding the first value at the end
df_values += df_values[:1]
df1_values += df1_values[:1]
angles = np.linspace(0, 2 * np.pi, len(categories), endpoint=False).tolist()
angles += angles[:1]

fig, ax = plt.subplots(figsize=(6, 6), subplot_kw=dict(polar=True))
ax.fill(angles, df_values, color='blue', alpha=0.25, label='df')
ax.plot(angles, df_values, color='blue', linewidth=2)

ax.fill(angles, df1_values, color='orange', alpha=0.25, label='df1')
ax.plot(angles, df1_values, color='orange', linewidth=2)

ax.set_yticklabels([])
ax.set_xticks(angles[:-1])
ax.set_xticklabels(categories)
plt.title('Radar Chart Comparing First Employee in df and df1')
plt.legend()
plt.show()
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Unlocking Insights: A Holistic Data Analysis Approach Using Advanced Statistical Techniques and Visualizations

In [ ]:
```
import pandas as pd  # For data manipulation and analysis
import numpy as np  # For numerical computations
import matplotlib.pyplot as plt  # For creating static, animated, and interactive visualizations
import seaborn as sns  # For statistical data visualization based on Matplotlib
import scipy  # For scientific and technical computing (including optimization, integration, and statistics)
from sklearn.preprocessing import StandardScaler, LabelEncoder  # For preprocessing data (scaling, encoding)
from sklearn.model_selection import train_test_split  # For splitting data into training and testing sets
from sklearn.linear_model import LinearRegression  # For linear regression models
from sklearn.metrics import mean_squared_error, r2_score  # For model evaluation metrics
import statsmodels.api as sm  # For statistical modeling and hypothesis testing
```

In [110…
```
ser = pd.Series(np.random.rand(34))
```

In [111…
```
type(ser)
```

Out[111… pandas.core.series.Series

In [112…
```
newdf = pd.DataFrame(np.random.rand(3343,20), index=np.arange(3343))
```

In [113…
```
newdf
```

Out[113…

	0	1	2	3	4	5	6	7	8	9	10	11	
0	0.697938	0.561076	0.001192	0.450341	0.795850	0.930623	0.767605	0.894239	0.597057	0.125559	0.751090	0.278110	0.45
1	0.046056	0.254787	0.427195	0.969418	0.213716	0.912283	0.974008	0.314478	0.691099	0.603735	0.979347	0.785053	0.3
2	0.854784	0.793532	0.031116	0.471809	0.772358	0.447498	0.452217	0.644761	0.218107	0.125786	0.519473	0.229090	0.93
3	0.562486	0.790980	0.744740	0.238140	0.171643	0.136453	0.936933	0.931844	0.380153	0.264501	0.673073	0.232032	0.12
4	0.702867	0.270176	0.268086	0.076133	0.174052	0.712256	0.812254	0.648388	0.217212	0.763016	0.771966	0.055136	0.15
...	...	...	...	...	...	...	...	...	...	...	...	...	
3338	0.242589	0.189695	0.527391	0.300127	0.902239	0.026674	0.125397	0.918755	0.432655	0.011054	0.326422	0.905210	0.2
3339	0.887061	0.909481	0.001816	0.454096	0.124479	0.180765	0.284239	0.307691	0.909599	0.909509	0.890847	0.829765	0.06
3340	0.531965	0.708004	0.179359	0.905226	0.234289	0.676948	0.990071	0.162226	0.992508	0.565066	0.343011	0.244501	0.4
3341	0.060029	0.033379	0.814435	0.271410	0.507328	0.512107	0.554627	0.549610	0.743996	0.252561	0.118287	0.719574	0.23
3342	0.930778	0.048853	0.275265	0.415583	0.517176	0.010622	0.092440	0.338265	0.599631	0.229081	0.001652	0.961243	0.28

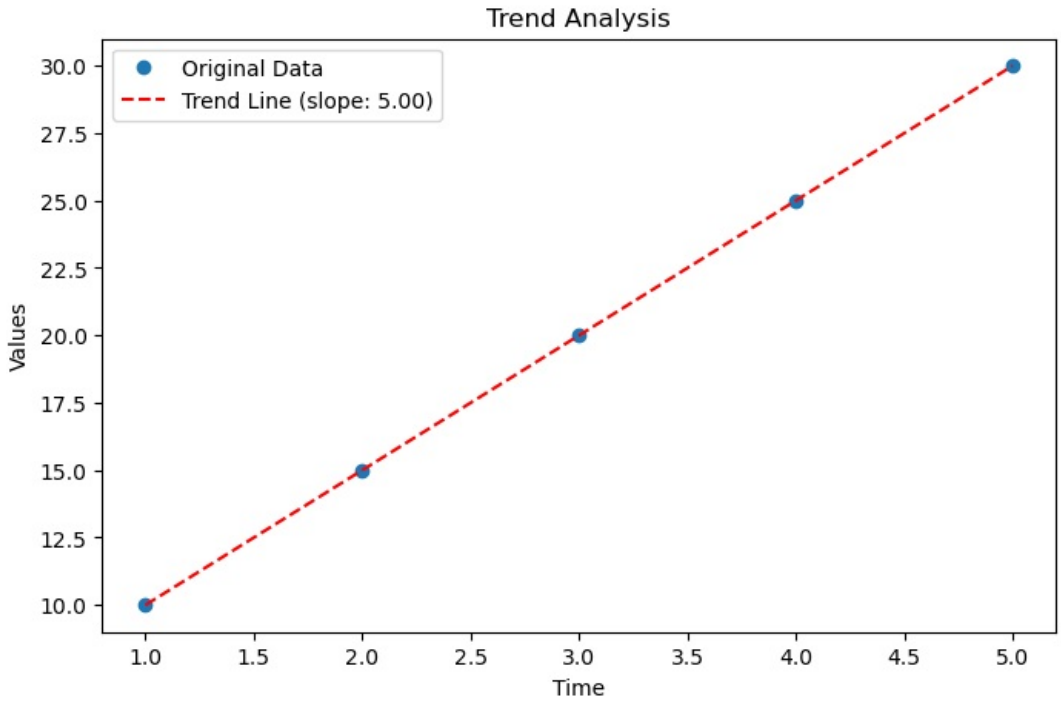
3343 rows × 20 columns

In [149…
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Sample Data Generation (Replace with your actual data loading)
# For demonstration purposes, let's assume we have some sample data
data = {
    'category': ['A', 'B', 'C', 'D'],
    'values': [25, 37, 50, 23],
    'errors': [2, 3, 4, 5]
}

df = pd.DataFrame(data)
```

Sample Data Generation (Replace with actual data loading)

In [157…
```
# Sample data
time_series = np.array([1, 2, 3, 4, 5])
values = np.array([10, 15, 20, 25, 30])

# Trend Analysis using Polyfit (linear trend)
trend = np.polyfit(time_series, values, 1)
trend_line = np.polyval(trend, time_series)

# Plotting the original data and trend line
plt.figure(figsize=(8, 5))
plt.plot(time_series, values, 'o', label='Original Data')
```

```
plt.plot(time_series, trend_line, 'r--', label=f'Trend Line (slope: {trend[0]:.2f})')
plt.title('Trend Analysis')
plt.xlabel('Time')
plt.ylabel('Values')
plt.legend()
plt.show()
```

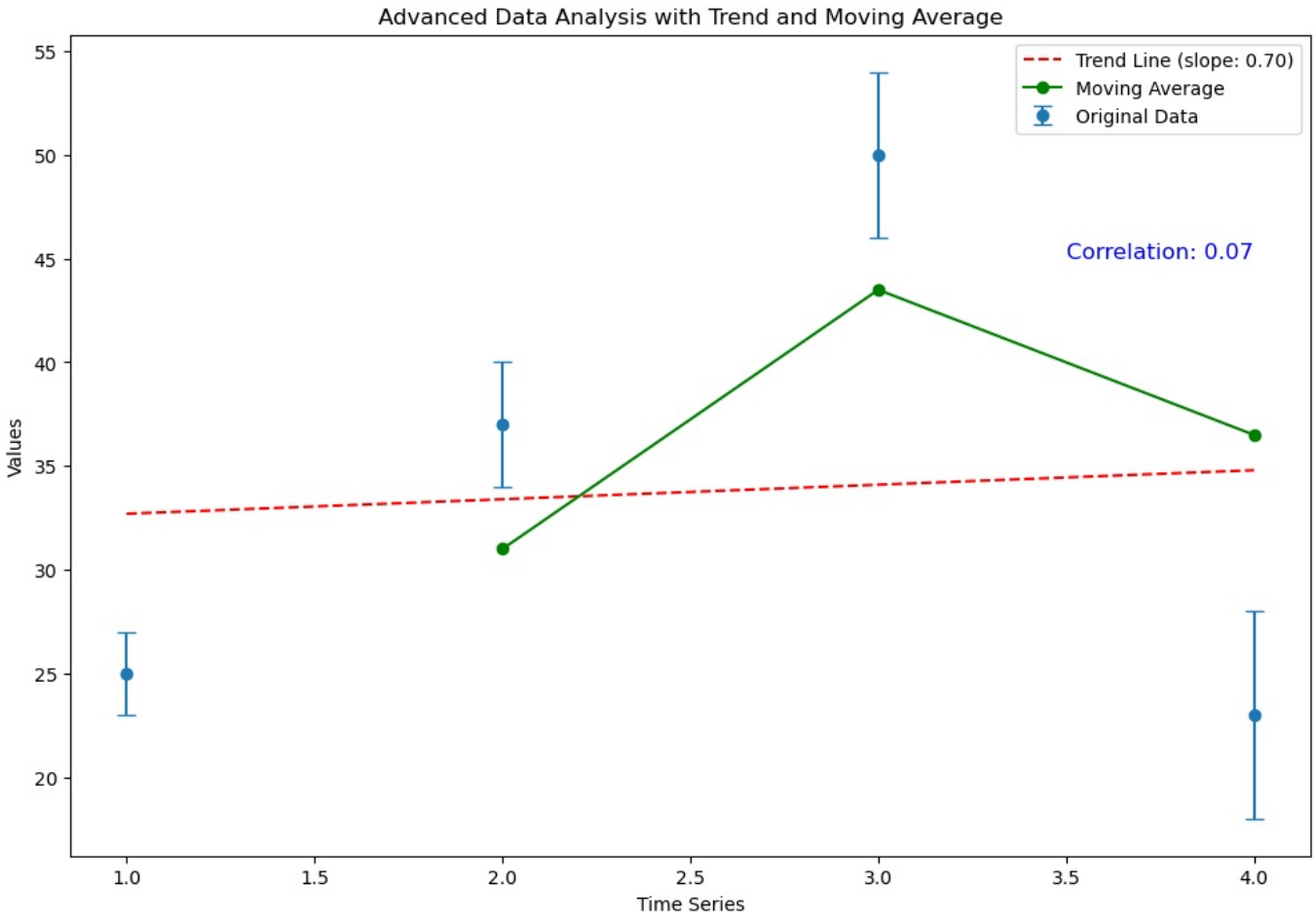


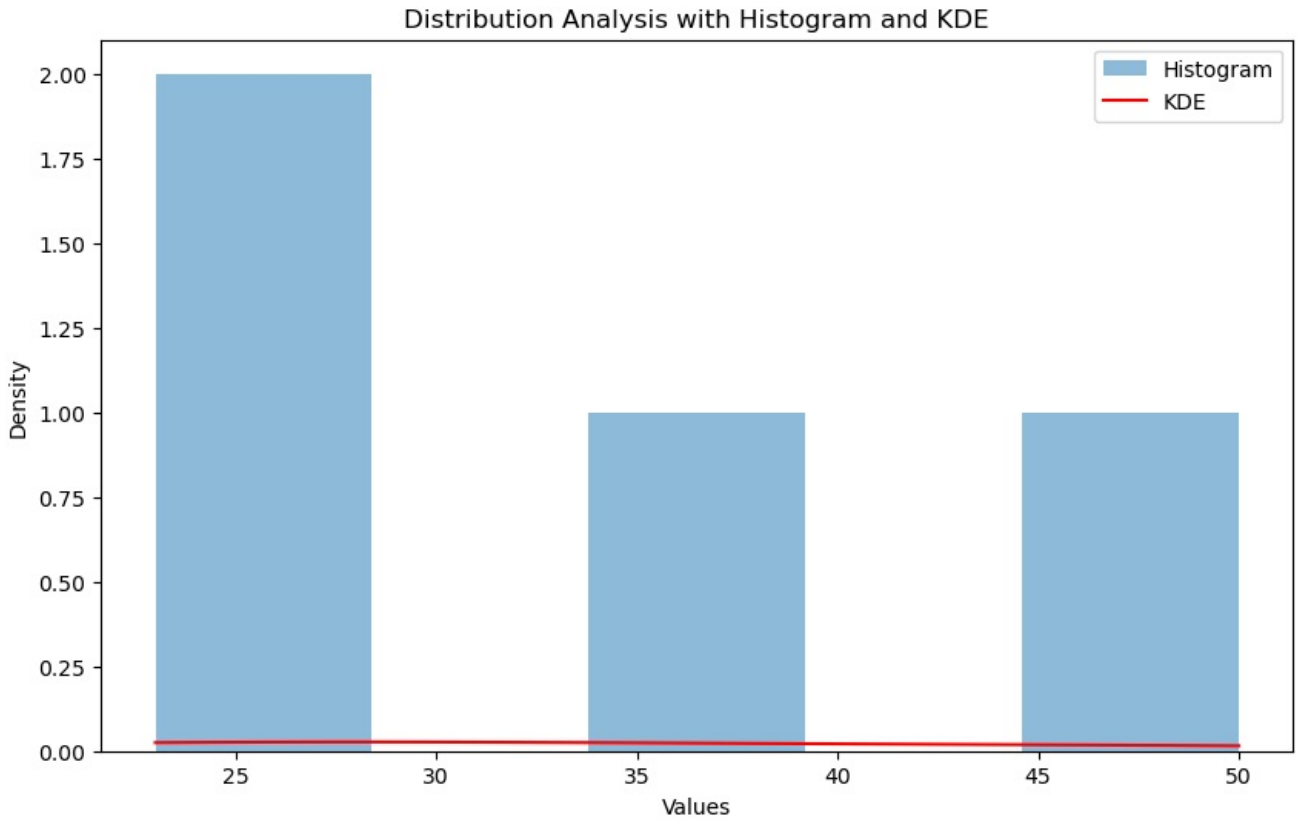
In [156…

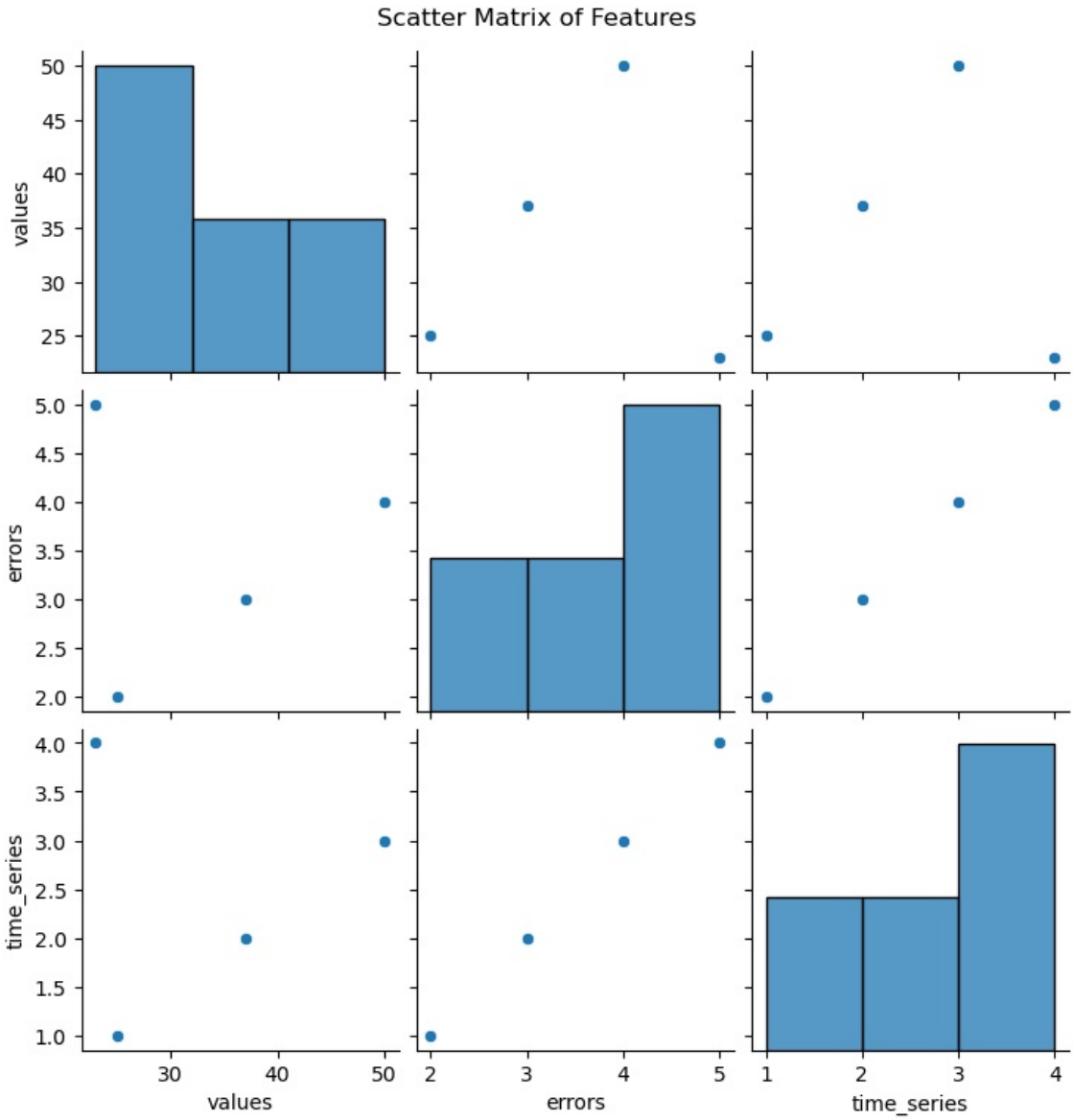
```
# This time, let's assume we have time-series data
data = {
    'category': ['A', 'B', 'C', 'D'],
    'values': [25, 37, 50, 23],
    'errors': [2, 3, 4, 5],
    'time_series': [1, 2, 3, 4]
}

df = pd.DataFrame(data)

# Convert data to NumPy arrays for detailed analysis
values = np.array(df['values'])
errors = np.array(df['errors'])
time_series = np.array(df['time_series'])

# 1. Trend Analysis using Polyfit (Polynomial Fitting)
trend = np.polyfit(time_series, values, 1)  # Linear trend
trend_line = np.polyval(trend, time_series)

# 2. Correlation Calculation using NumPy
correlation = np.corrcoef(time_series, values)[0, 1]
print(f"Correlation between Time and Values: {correlation:.2f}")

# 3. Moving Average Calculation
window_size = 2
moving_avg = np.convolve(values, np.ones(window_size)/window_size, mode='valid')

# Visualization using Matplotlib

plt.figure(figsize=(12, 8))

# Original Data with Error Bars
plt.errorbar(df['time_series'], values, yerr=errors, fmt='o', label='Original Data', capsize=5)

# Trend Line
plt.plot(time_series, trend_line, label=f'Trend Line (slope: {trend[0]:.2f})', color='r', linestyle='--')

# Moving Average
plt.plot(time_series[window_size-1:], moving_avg, label='Moving Average', color='g', marker='o')

# Annotations for Correlation
plt.text(3.5, np.max(values) - 5, f'Correlation: {correlation:.2f}', fontsize=12, color='b')

# Adding titles and labels
plt.title('Advanced Data Analysis with Trend and Moving Average')
plt.xlabel('Time Series')
plt.ylabel('Values')
plt.legend()
```

```python
# Show plot
plt.show()

# 4. Histogram and KDE Plot for Distribution Analysis
plt.figure(figsize=(10, 6))

# Histogram
plt.hist(values, bins=5, alpha=0.5, label='Histogram')

# Kernel Density Estimation (KDE) using Matplotlib
from scipy.stats import gaussian_kde
kde = gaussian_kde(values)
kde_x = np.linspace(min(values), max(values), 100)
kde_y = kde(kde_x)

plt.plot(kde_x, kde_y, color='r', label='KDE')

# Adding titles and labels
plt.title('Distribution Analysis with Histogram and KDE')
plt.xlabel('Values')
plt.ylabel('Density')
plt.legend()

# Show plot
plt.show()

# 5. Scatter Matrix for Pairwise Relationships
import seaborn as sns

# Assuming the dataframe has multiple numerical columns
sns.pairplot(df)
plt.suptitle('Scatter Matrix of Features', y=1.02)

plt.show()
```

```
Correlation between Time and Values: 0.07
```




Distribution Analysis with Histogram and KDE
Histogram
KDE
Density
Values
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
25
30
35
40
45
50



In [148…
```
# Print calculated statistics
print(f"Mean: {mean}")
print(f"Standard Deviation: {std_dev}")
print(f"Variance: {variance}")
print(f"Median: {median}")
```

```
Mean: 33.75
Standard Deviation: 10.80219885023415
Variance: 116.6875
Median: 31.0
```

In [151…
```
# Convert data to NumPy arrays for detailed analysis
values = np.array(df['values'])
errors = np.array(df['errors'])

# Example of advanced analysis using NumPy
mean = np.mean(values)
std_dev = np.std(values)
variance = np.var(values)
median = np.median(values)
```

In [152…
```
print(newdf)
```

```
            0         1         2         3         4         5         6   \
0     satedner  0.561076  0.001192  0.450341  0.795850  0.930623  0.767605
1        Rahul  0.254787  0.427195  0.969418  0.213716  0.912283  0.974008
2     0.854784  0.793532  0.031116  0.471809  0.772358  0.447498  0.452217
3     0.562486  0.790980  0.744740  0.238140  0.171643  0.136453  0.936933
4     0.702867  0.270176  0.268086  0.076133  0.174052  0.712256  0.812254
...        ...       ...       ...       ...       ...       ...       ...
3338  0.242589  0.189695  0.527391  0.300127  0.902239  0.026674  0.125397
3339  0.887061  0.909481  0.001816  0.454096  0.124479  0.180765  0.284239
3340  0.531965  0.708004  0.179359  0.905226  0.234289  0.676948  0.990071
3341  0.060029  0.033379  0.814435  0.271410  0.507328  0.512107  0.554627
3342  0.930778  0.048853  0.275265  0.415583  0.517176  0.010622  0.092440

            7         8         9         10        11        12        13  \
0     0.894239  0.597057  0.125559  0.751090  0.278110  0.459299  0.171059
1     0.314478  0.691099  0.603735  0.979347  0.785053  0.310106  0.672351
2     0.644761  0.218107  0.125786  0.519473  0.229090  0.934016  0.107259
3     0.931844  0.380153  0.264501  0.673073  0.232032  0.122061  0.273927
4     0.648388  0.217212  0.763016  0.771966  0.055136  0.151104  0.818651
...        ...       ...       ...       ...       ...       ...       ...
3338  0.918755  0.432655  0.011054  0.326422  0.905210  0.211183  0.424995
3339  0.307691  0.909599  0.909509  0.890847  0.829765  0.066687  0.026478
3340  0.162226  0.992508  0.565066  0.343011  0.244501  0.411046  0.455853
3341  0.549610  0.743996  0.252561  0.118287  0.719574  0.237188  0.160270
3342  0.338265  0.599631  0.229081  0.001652  0.961243  0.287422  0.668404

            14        15        16        17        18        19
0     0.037895  0.065044  0.596998  0.741632  0.391194  0.382090
1     0.709704  0.082982  0.125456  0.143640  0.509255  0.451089
2     0.003470  0.085419  0.023010  0.987459  0.096069  0.430543
3     0.399441  0.192537  0.233651  0.729097  0.557619  0.680914
4     0.847048  0.466855  0.159128  0.219528  0.235784  0.922468
...        ...       ...       ...       ...       ...       ...
3338  0.089949  0.718238  0.329754  0.482533  0.567450  0.358958
3339  0.136172  0.921743  0.742154  0.520271  0.610764  0.880770
3340  0.369144  0.977681  0.170403  0.202506  0.942130  0.444679
3341  0.124291  0.142488  0.355018  0.010618  0.548974  0.066305
3342  0.187772  0.290403  0.631293  0.002956  0.535297  0.905234

[3343 rows x 20 columns]
```

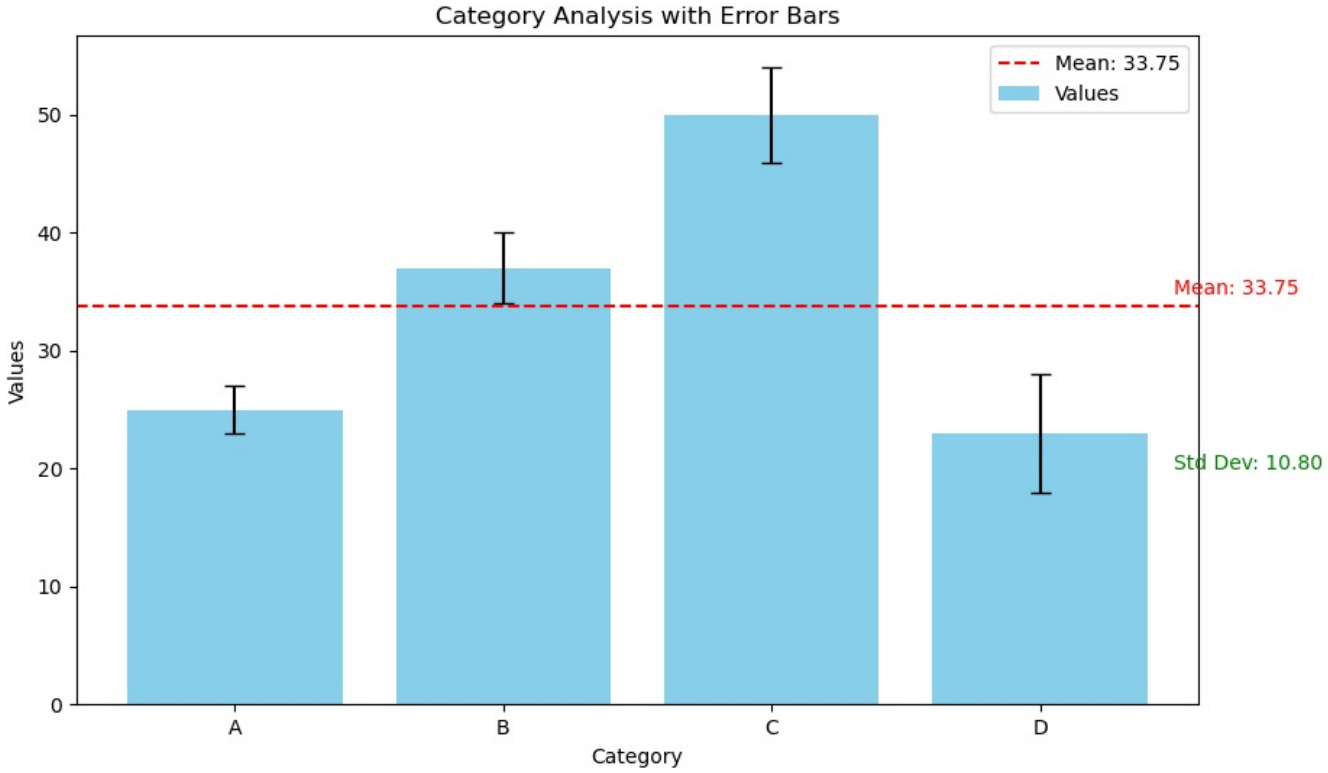
Print calculated statistics

In [154…

```
print(f"Mean: {mean}")
print(f"Standard Deviation: {std_dev}")
print(f"Variance: {variance}")
print(f"Median: {median}")

# Visualization using Matplotlib
plt.figure(figsize=(10, 6))

# Bar plot with error bars
plt.bar(df['category'], values, yerr=errors, capsize=5, color='skyblue', label='Values')

# Add mean line
plt.axhline(y=mean, color='r', linestyle='--', label=f'Mean: {mean:.2f}')

# Add text annotations for statistics
plt.text(3.5, mean + 1, f'Mean: {mean:.2f}', color='r')
plt.text(3.5, mean - std_dev - 3, f'Std Dev: {std_dev:.2f}', color='g')
plt.text(3.5, mean - variance - 6, f'Variance: {variance:.2f}', color='b')

# Adding titles and labels
plt.title('Category Analysis with Error Bars')
plt.xlabel('Category')
plt.ylabel('Values')
plt.legend()

# Show plot
plt.show()

# Advanced NumPy operations: Normalizing the data
normalized_values = (values - mean) / std_dev
print("Normalized Values:", normalized_values)

# Visualization of normalized data
plt.figure(figsize=(10, 6))
plt.bar(df['category'], normalized_values, color='orange', label='Normalized Values')

# Add titles and labels
plt.title('Normalized Values by Category')
plt.xlabel('Category')
plt.ylabel('Normalized Value')
plt.legend()
```

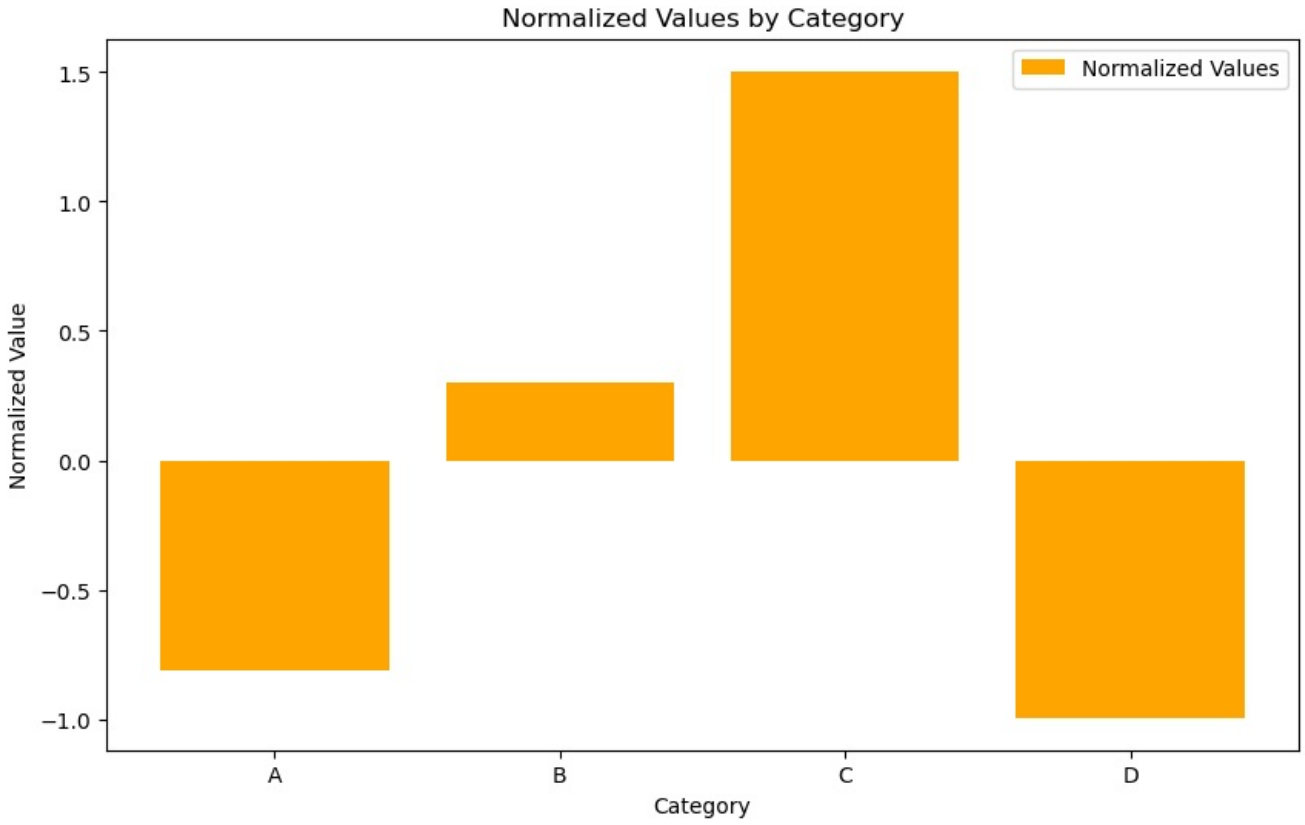
```
# Show plot
plt.show()
```

```
Mean: 33.75
Standard Deviation: 10.80219885023415
Variance: 116.6875
Median: 31.0
```



Variance: 116.69

```
Normalized Values: [-0.81002027  0.30086467  1.50432335 -0.99516776]
```



In [114…
```
newdf.head()
```

Out[114…

	0	1	2	3	4	5	6	7	8	9	10	11	1
0	0.697938	0.561076	0.001192	0.450341	0.795850	0.930623	0.767605	0.894239	0.597057	0.125559	0.751090	0.278110	0.45929
1	0.046056	0.254787	0.427195	0.969418	0.213716	0.912283	0.974008	0.314478	0.691099	0.603735	0.979347	0.785053	0.31010
2	0.854784	0.793532	0.031116	0.471809	0.772358	0.447498	0.452217	0.644761	0.218107	0.125786	0.519473	0.229090	0.93401
3	0.562486	0.790980	0.744740	0.238140	0.171643	0.136453	0.936933	0.931844	0.380153	0.264501	0.673073	0.232032	0.12206
4	0.702867	0.270176	0.268086	0.076133	0.174052	0.712256	0.812254	0.648388	0.217212	0.763016	0.771966	0.055136	0.15110

In [115…
```
type(newdf)
```

Out[115… `pandas.core.frame.DataFrame`

In [116…
```
newdf.describe()
```

Out[116…

	0	1	2	3	4	5	6	7	8	
count	3343.000000	3343.000000	3343.000000	3343.000000	3343.000000	3343.000000	3343.000000	3343.000000	3343.000000	334
mean	0.503739	0.491004	0.491591	0.501394	0.493043	0.505059	0.505016	0.503800	0.505401	
std	0.289272	0.288965	0.287712	0.286441	0.289059	0.291096	0.290702	0.291508	0.283214	
min	0.000187	0.000288	0.000137	0.000232	0.000127	0.000114	0.000035	0.000412	0.000371	
25%	0.258086	0.232902	0.241878	0.255746	0.245966	0.249273	0.253529	0.252640	0.265151	
50%	0.509689	0.490917	0.487950	0.502484	0.485434	0.510141	0.511370	0.509279	0.505908	
75%	0.754876	0.742270	0.741858	0.748114	0.742720	0.754881	0.763551	0.760554	0.745003	
max	0.999349	0.999383	0.999779	0.999549	0.999999	0.999499	0.999996	0.999889	0.999710	

In [117…
```
newdf.dtypes
```

Out[117…
```
0     float64
1     float64
2     float64
3     float64
4     float64
5     float64
6     float64
7     float64
8     float64
9     float64
10    float64
11    float64
12    float64
13    float64
14    float64
15    float64
16    float64
17    float64
18    float64
19    float64
dtype: object
```

In [120…
```
# string put and change
```

In [121…
```
newdf.head()
```

Out[121…

	0	1	2	3	4	5	6	7	8	9	10	11	1
0	satedner	0.561076	0.001192	0.450341	0.795850	0.930623	0.767605	0.894239	0.597057	0.125559	0.751090	0.278110	0.45929
1	0.046056	0.254787	0.427195	0.969418	0.213716	0.912283	0.974008	0.314478	0.691099	0.603735	0.979347	0.785053	0.3101(
2	0.854784	0.793532	0.031116	0.471809	0.772358	0.447498	0.452217	0.644761	0.218107	0.125786	0.519473	0.229090	0.9340
3	0.562486	0.790980	0.744740	0.238140	0.171643	0.136453	0.936933	0.931844	0.380153	0.264501	0.673073	0.232032	0.1220(
4	0.702867	0.270176	0.268086	0.076133	0.174052	0.712256	0.812254	0.648388	0.217212	0.763016	0.771966	0.055136	0.1511(

In [122…
```
newdf.index
```

Out[122…
```
Index([   0,    1,    2,    3,    4,    5,    6,    7,    8,    9,
       ...
       3333, 3334, 3335, 3336, 3337, 3338, 3339, 3340, 3341, 3342],
      dtype='int32', length=3343)
```

In [123…
```
newdf.columns
```

Out[123…
```
RangeIndex(start=0, stop=20, step=1)
```

In [124…
```
newdf.to_numpy()
```

Out[124…
```
array([['satedner', 0.5610761589842846, 0.001192114587430293, ...,
        0.7416324294616016, 0.3911937244361793, 0.3820896402352686],
       [0.046055937342585174, 0.254787362097771, 0.42719452008559466,
        ..., 0.14364003953672966, 0.5092546822021528, 0.4510891827054363],
       [0.8547835023621424, 0.7935318872325342, 0.03111625504524096, ...,
        0.987459426960006, 0.09606897531240688, 0.43054322731525496],
       ...,
       [0.5319647386673982, 0.7080038836479031, 0.17935934394912378, ...,
        0.2025064014616803, 0.9421295704371534, 0.4446792476788721],
       [0.060028950655676594, 0.03337944486345812, 0.814435289632806,
        ..., 0.010618131787275442, 0.5489742252985179,
        0.06630462134075654],
       [0.9307781597401553, 0.04885281230366967, 0.2752652403856555, ...,
        0.002956251639411489, 0.5352968436617144, 0.9052336521663139]],
      dtype=object)
```

In [125…
```
newdf.to_numpy()
```

Out[125…
```
array([['satedner', 0.5610761589842846, 0.001192114587430293, ...,
        0.7416324294616016, 0.3911937244361793, 0.3820896402352686],
       [0.046055937342585174, 0.254787362097771, 0.42719452008559466,
        ..., 0.14364003953672966, 0.5092546822021528, 0.4510891827054363],
       [0.8547835023621424, 0.7935318872325342, 0.03111625504524096, ...,
        0.987459426960006, 0.09606897531240688, 0.43054322731525496],
       ...,
       [0.5319647386673982, 0.7080038836479031, 0.17935934394912378, ...,
        0.2025064014616803, 0.9421295704371534, 0.4446792476788721],
       [0.060028950655676594, 0.03337944486345812, 0.814435289632806,
        ..., 0.010618131787275442, 0.5489742252985179,
        0.06630462134075654],
       [0.9307781597401553, 0.04885281230366967, 0.2752652403856555, ...,
        0.002956251639411489, 0.5352968436617144, 0.9052336521663139]],
      dtype=object)
```

In [126…
```
newdf.T
```

Out[126…

	0	1	2	3	4	5	6	7	8	9	...	3333	3334	
0	satedner	0.046056	0.854784	0.562486	0.702867	0.95931	0.024801	0.01371	0.793867	0.763245	...	0.704641	0.790551	0.
1	0.561076	0.254787	0.793532	0.79098	0.270176	0.940277	0.527591	0.241154	0.863165	0.120385	...	0.024275	0.678563	0.
2	0.001192	0.427195	0.031116	0.74474	0.268086	0.411395	0.615717	0.916726	0.084959	0.306412	...	0.155299	0.001609	0.
3	0.450341	0.969418	0.471809	0.23814	0.076133	0.207079	0.650445	0.498439	0.524331	0.414657	...	0.945955	0.546366	0.
4	0.79585	0.213716	0.772358	0.171643	0.174052	0.966039	0.54581	0.639105	0.250488	0.946433	...	0.024077	0.238793	0.
5	0.930623	0.912283	0.447498	0.136453	0.712256	0.672888	0.432509	0.16736	0.097094	0.225367	...	0.945789	0.88016	0
6	0.767605	0.974008	0.452217	0.936933	0.812254	0.175457	0.530555	0.812364	0.481824	0.303791	...	0.189091	0.665103	0.
7	0.894239	0.314478	0.644761	0.931844	0.648388	0.859932	0.942829	0.088588	0.859426	0.758608	...	0.428781	0.158825	0.
8	0.597057	0.691099	0.218107	0.380153	0.217212	0.902214	0.769244	0.720237	0.132585	0.048448	...	0.594817	0.221407	0.
9	0.125559	0.603735	0.125786	0.264501	0.763016	0.452349	0.999601	0.576733	0.412966	0.371679	...	0.415409	0.531016	0.
10	0.75109	0.979347	0.519473	0.673073	0.771966	0.565652	0.403283	0.669826	0.737933	0.70599	...	0.473757	0.329678	0.
11	0.27811	0.785053	0.22909	0.232032	0.055136	0.911236	0.617311	0.338826	0.633086	0.012504	...	0.822269	0.301242	0.
12	0.459299	0.310106	0.934016	0.122061	0.151104	0.498092	0.357427	0.603305	0.112751	0.385815	...	0.138073	0.008323	0.
13	0.171059	0.672351	0.107259	0.273927	0.818651	0.641221	0.162805	0.180483	0.963244	0.947437	...	0.779108	0.481304	0.
14	0.037895	0.709704	0.00347	0.399441	0.847048	0.281745	0.223323	0.999188	0.025439	0.420454	...	0.372698	0.708471	0.
15	0.065044	0.082982	0.085419	0.192537	0.466855	0.687869	0.025836	0.74007	0.53407	0.828131	...	0.324419	0.228337	0
16	0.596998	0.125456	0.02301	0.233651	0.159128	0.2913	0.774809	0.599471	0.631992	0.013455	...	0.500413	0.180724	0.
17	0.741632	0.14364	0.987459	0.729097	0.219528	0.233315	0.391999	0.007596	0.069191	0.946884	...	0.754954	0.28647	0.
18	0.391194	0.509255	0.096069	0.557619	0.235784	0.311641	0.607267	0.936231	0.640682	0.851307	...	0.092456	0.156106	0.
19	0.38209	0.451089	0.430543	0.680914	0.922468	0.441176	0.553356	0.879719	0.612646	0.933405	...	0.17309	0.962706	0.

20 rows × 3343 columns

In [127…
```
newdf
```

Out[127…

	0	1	2	3	4	5	6	7	8	9	10	11	
0	satedner	0.561076	0.001192	0.450341	0.795850	0.930623	0.767605	0.894239	0.597057	0.125559	0.751090	0.278110	0.45
1	0.046056	0.254787	0.427195	0.969418	0.213716	0.912283	0.974008	0.314478	0.691099	0.603735	0.979347	0.785053	0.3
2	0.854784	0.793532	0.031116	0.471809	0.772358	0.447498	0.452217	0.644761	0.218107	0.125786	0.519473	0.229090	0.93
3	0.562486	0.790980	0.744740	0.238140	0.171643	0.136453	0.936933	0.931844	0.380153	0.264501	0.673073	0.232032	0.12
4	0.702867	0.270176	0.268086	0.076133	0.174052	0.712256	0.812254	0.648388	0.217212	0.763016	0.771966	0.055136	0.15
...	...	...	...	...	...	...	...	...	...	...	...	...	...
3338	0.242589	0.189695	0.527391	0.300127	0.902239	0.026674	0.125397	0.918755	0.432655	0.011054	0.326422	0.905210	0.2
3339	0.887061	0.909481	0.001816	0.454096	0.124479	0.180765	0.284239	0.307691	0.909599	0.909509	0.890847	0.829765	0.06
3340	0.531965	0.708004	0.179359	0.905226	0.234289	0.676948	0.990071	0.162226	0.992508	0.565066	0.343011	0.244501	0.4
3341	0.060029	0.033379	0.814435	0.271410	0.507328	0.512107	0.554627	0.549610	0.743996	0.252561	0.118287	0.719574	0.23
3342	0.930778	0.048853	0.275265	0.415583	0.517176	0.010622	0.092440	0.338265	0.599631	0.229081	0.001652	0.961243	0.28

3343 rows × 20 columns

In [128…
```
newdf.head()
```

Out[128…

	0	1	2	3	4	5	6	7	8	9	10	11	1
0	satedner	0.561076	0.001192	0.450341	0.795850	0.930623	0.767605	0.894239	0.597057	0.125559	0.751090	0.278110	0.45929
1	0.046056	0.254787	0.427195	0.969418	0.213716	0.912283	0.974008	0.314478	0.691099	0.603735	0.979347	0.785053	0.3101(
2	0.854784	0.793532	0.031116	0.471809	0.772358	0.447498	0.452217	0.644761	0.218107	0.125786	0.519473	0.229090	0.9340
3	0.562486	0.790980	0.744740	0.238140	0.171643	0.136453	0.936933	0.931844	0.380153	0.264501	0.673073	0.232032	0.1220(
4	0.702867	0.270176	0.268086	0.076133	0.174052	0.712256	0.812254	0.648388	0.217212	0.763016	0.771966	0.055136	0.1511(

In [129…
```
newdf.sort_index()
```

Out[129…

	0	1	2	3	4	5	6	7	8	9	10	11	
0	satedner	0.561076	0.001192	0.450341	0.795850	0.930623	0.767605	0.894239	0.597057	0.125559	0.751090	0.278110	0.45
1	0.046056	0.254787	0.427195	0.969418	0.213716	0.912283	0.974008	0.314478	0.691099	0.603735	0.979347	0.785053	0.3
2	0.854784	0.793532	0.031116	0.471809	0.772358	0.447498	0.452217	0.644761	0.218107	0.125786	0.519473	0.229090	0.93
3	0.562486	0.790980	0.744740	0.238140	0.171643	0.136453	0.936933	0.931844	0.380153	0.264501	0.673073	0.232032	0.12
4	0.702867	0.270176	0.268086	0.076133	0.174052	0.712256	0.812254	0.648388	0.217212	0.763016	0.771966	0.055136	0.15
...	...	...	...	...	...	...	...	...	...	...	...	...	
3338	0.242589	0.189695	0.527391	0.300127	0.902239	0.026674	0.125397	0.918755	0.432655	0.011054	0.326422	0.905210	0.2
3339	0.887061	0.909481	0.001816	0.454096	0.124479	0.180765	0.284239	0.307691	0.909599	0.909509	0.890847	0.829765	0.06
3340	0.531965	0.708004	0.179359	0.905226	0.234289	0.676948	0.990071	0.162226	0.992508	0.565066	0.343011	0.244501	0.4
3341	0.060029	0.033379	0.814435	0.271410	0.507328	0.512107	0.554627	0.549610	0.743996	0.252561	0.118287	0.719574	0.23
3342	0.930778	0.048853	0.275265	0.415583	0.517176	0.010622	0.092440	0.338265	0.599631	0.229081	0.001652	0.961243	0.28

3343 rows × 20 columns

In [130…
```
newdf
```

Out[130…

	0	1	2	3	4	5	6	7	8	9	10	11	
0	satedner	0.561076	0.001192	0.450341	0.795850	0.930623	0.767605	0.894239	0.597057	0.125559	0.751090	0.278110	0.45
1	0.046056	0.254787	0.427195	0.969418	0.213716	0.912283	0.974008	0.314478	0.691099	0.603735	0.979347	0.785053	0.3
2	0.854784	0.793532	0.031116	0.471809	0.772358	0.447498	0.452217	0.644761	0.218107	0.125786	0.519473	0.229090	0.93
3	0.562486	0.790980	0.744740	0.238140	0.171643	0.136453	0.936933	0.931844	0.380153	0.264501	0.673073	0.232032	0.12
4	0.702867	0.270176	0.268086	0.076133	0.174052	0.712256	0.812254	0.648388	0.217212	0.763016	0.771966	0.055136	0.15
...	...	...	...	...	...	...	...	...	...	...	...	...	
3338	0.242589	0.189695	0.527391	0.300127	0.902239	0.026674	0.125397	0.918755	0.432655	0.011054	0.326422	0.905210	0.2
3339	0.887061	0.909481	0.001816	0.454096	0.124479	0.180765	0.284239	0.307691	0.909599	0.909509	0.890847	0.829765	0.06
3340	0.531965	0.708004	0.179359	0.905226	0.234289	0.676948	0.990071	0.162226	0.992508	0.565066	0.343011	0.244501	0.4
3341	0.060029	0.033379	0.814435	0.271410	0.507328	0.512107	0.554627	0.549610	0.743996	0.252561	0.118287	0.719574	0.23
3342	0.930778	0.048853	0.275265	0.415583	0.517176	0.010622	0.092440	0.338265	0.599631	0.229081	0.001652	0.961243	0.28

3343 rows × 20 columns

In [131…
```
newdf
```

Out[131…

	0	1	2	3	4	5	6	7	8	9	10	11	
0	satedner	0.561076	0.001192	0.450341	0.795850	0.930623	0.767605	0.894239	0.597057	0.125559	0.751090	0.278110	0.45
1	0.046056	0.254787	0.427195	0.969418	0.213716	0.912283	0.974008	0.314478	0.691099	0.603735	0.979347	0.785053	0.3
2	0.854784	0.793532	0.031116	0.471809	0.772358	0.447498	0.452217	0.644761	0.218107	0.125786	0.519473	0.229090	0.93
3	0.562486	0.790980	0.744740	0.238140	0.171643	0.136453	0.936933	0.931844	0.380153	0.264501	0.673073	0.232032	0.12
4	0.702867	0.270176	0.268086	0.076133	0.174052	0.712256	0.812254	0.648388	0.217212	0.763016	0.771966	0.055136	0.15
...	...	...	...	...	...	...	...	...	...	...	...	...	
3338	0.242589	0.189695	0.527391	0.300127	0.902239	0.026674	0.125397	0.918755	0.432655	0.011054	0.326422	0.905210	0.2
3339	0.887061	0.909481	0.001816	0.454096	0.124479	0.180765	0.284239	0.307691	0.909599	0.909509	0.890847	0.829765	0.06
3340	0.531965	0.708004	0.179359	0.905226	0.234289	0.676948	0.990071	0.162226	0.992508	0.565066	0.343011	0.244501	0.4
3341	0.060029	0.033379	0.814435	0.271410	0.507328	0.512107	0.554627	0.549610	0.743996	0.252561	0.118287	0.719574	0.23
3342	0.930778	0.048853	0.275265	0.415583	0.517176	0.010622	0.092440	0.338265	0.599631	0.229081	0.001652	0.961243	0.28

3343 rows × 20 columns

In [132…
```
# Numbering by data in DataFrame - SK
```

In [133…
```
newdf[0]
```

Out[133…
```
0       satedner
1       0.046056
2       0.854784
3       0.562486
4       0.702867
          ...
3338    0.242589
3339    0.887061
3340    0.531965
3341    0.060029
3342    0.930778
Name: 0, Length: 3343, dtype: object
```

In [134…
```
newdf[1]
```

Out[134…
```
0       0.561076
1       0.254787
2       0.793532
3       0.790980
4       0.270176
          ...
3338    0.189695
3339    0.909481
3340    0.708004
3341    0.033379
3342    0.048853
Name: 1, Length: 3343, dtype: float64
```

In [135…
```
newdf[3]
```

Out[135…
```
0       0.450341
1       0.969418
2       0.471809
3       0.238140
4       0.076133
          ...
3338    0.300127
3339    0.454096
3340    0.905226
3341    0.271410
3342    0.415583
Name: 3, Length: 3343, dtype: float64
```

In [136…
```
newdf[4]
```

Out[136…
```
0       0.795850
1       0.213716
2       0.772358
3       0.171643
4       0.174052
          ...
3338    0.902239
3339    0.124479
3340    0.234289
3341    0.507328
3342    0.517176
Name: 4, Length: 3343, dtype: float64
```

In [137…
```
newdf
```

Out[137…

	0	1	2	3	4	5	6	7	8	9	10	11	
0	satedner	0.561076	0.001192	0.450341	0.795850	0.930623	0.767605	0.894239	0.597057	0.125559	0.751090	0.278110	0.45
1	0.046056	0.254787	0.427195	0.969418	0.213716	0.912283	0.974008	0.314478	0.691099	0.603735	0.979347	0.785053	0.3
2	0.854784	0.793532	0.031116	0.471809	0.772358	0.447498	0.452217	0.644761	0.218107	0.125786	0.519473	0.229090	0.93
3	0.562486	0.790980	0.744740	0.238140	0.171643	0.136453	0.936933	0.931844	0.380153	0.264501	0.673073	0.232032	0.12
4	0.702867	0.270176	0.268086	0.076133	0.174052	0.712256	0.812254	0.648388	0.217212	0.763016	0.771966	0.055136	0.15
...	...	...	...	...	...	...	...	...	...	...	...	...	
3338	0.242589	0.189695	0.527391	0.300127	0.902239	0.026674	0.125397	0.918755	0.432655	0.011054	0.326422	0.905210	0.2
3339	0.887061	0.909481	0.001816	0.454096	0.124479	0.180765	0.284239	0.307691	0.909599	0.909509	0.890847	0.829765	0.06
3340	0.531965	0.708004	0.179359	0.905226	0.234289	0.676948	0.990071	0.162226	0.992508	0.565066	0.343011	0.244501	0.4
3341	0.060029	0.033379	0.814435	0.271410	0.507328	0.512107	0.554627	0.549610	0.743996	0.252561	0.118287	0.719574	0.23
3342	0.930778	0.048853	0.275265	0.415583	0.517176	0.010622	0.092440	0.338265	0.599631	0.229081	0.001652	0.961243	0.28

3343 rows × 20 columns

In [138…
```
newdf
```

Out[138…

	0	1	2	3	4	5	6	7	8	9	10	11	
0	satedner	0.561076	0.001192	0.450341	0.795850	0.930623	0.767605	0.894239	0.597057	0.125559	0.751090	0.278110	0.45
1	0.046056	0.254787	0.427195	0.969418	0.213716	0.912283	0.974008	0.314478	0.691099	0.603735	0.979347	0.785053	0.3
2	0.854784	0.793532	0.031116	0.471809	0.772358	0.447498	0.452217	0.644761	0.218107	0.125786	0.519473	0.229090	0.93
3	0.562486	0.790980	0.744740	0.238140	0.171643	0.136453	0.936933	0.931844	0.380153	0.264501	0.673073	0.232032	0.12
4	0.702867	0.270176	0.268086	0.076133	0.174052	0.712256	0.812254	0.648388	0.217212	0.763016	0.771966	0.055136	0.15
...	...	...	...	...	...	...	...	...	...	...	...	...	
3338	0.242589	0.189695	0.527391	0.300127	0.902239	0.026674	0.125397	0.918755	0.432655	0.011054	0.326422	0.905210	0.2
3339	0.887061	0.909481	0.001816	0.454096	0.124479	0.180765	0.284239	0.307691	0.909599	0.909509	0.890847	0.829765	0.06
3340	0.531965	0.708004	0.179359	0.905226	0.234289	0.676948	0.990071	0.162226	0.992508	0.565066	0.343011	0.244501	0.4
3341	0.060029	0.033379	0.814435	0.271410	0.507328	0.512107	0.554627	0.549610	0.743996	0.252561	0.118287	0.719574	0.23
3342	0.930778	0.048853	0.275265	0.415583	0.517176	0.010622	0.092440	0.338265	0.599631	0.229081	0.001652	0.961243	0.28

3343 rows × 20 columns

In [139…
```
newdf.loc[1,0] = "Rahul"
```

In [140…
```
newdf
```

Out[140…

	0	1	2	3	4	5	6	7	8	9	10	11	
0	satedner	0.561076	0.001192	0.450341	0.795850	0.930623	0.767605	0.894239	0.597057	0.125559	0.751090	0.278110	0.45
1	Rahul	0.254787	0.427195	0.969418	0.213716	0.912283	0.974008	0.314478	0.691099	0.603735	0.979347	0.785053	0.3
2	0.854784	0.793532	0.031116	0.471809	0.772358	0.447498	0.452217	0.644761	0.218107	0.125786	0.519473	0.229090	0.93
3	0.562486	0.790980	0.744740	0.238140	0.171643	0.136453	0.936933	0.931844	0.380153	0.264501	0.673073	0.232032	0.12
4	0.702867	0.270176	0.268086	0.076133	0.174052	0.712256	0.812254	0.648388	0.217212	0.763016	0.771966	0.055136	0.15
...	...	...	...	...	...	...	...	...	...	...	...	...	
3338	0.242589	0.189695	0.527391	0.300127	0.902239	0.026674	0.125397	0.918755	0.432655	0.011054	0.326422	0.905210	0.2
3339	0.887061	0.909481	0.001816	0.454096	0.124479	0.180765	0.284239	0.307691	0.909599	0.909509	0.890847	0.829765	0.06
3340	0.531965	0.708004	0.179359	0.905226	0.234289	0.676948	0.990071	0.162226	0.992508	0.565066	0.343011	0.244501	0.4
3341	0.060029	0.033379	0.814435	0.271410	0.507328	0.512107	0.554627	0.549610	0.743996	0.252561	0.118287	0.719574	0.23
3342	0.930778	0.048853	0.275265	0.415583	0.517176	0.010622	0.092440	0.338265	0.599631	0.229081	0.001652	0.961243	0.28

3343 rows × 20 columns

In [141…
```
type(newdf)
```

Out[141… pandas.core.frame.DataFrame

In [143…
```
newdf
```

Out[143…

	0	1	2	3	4	5	6	7	8	9	10	11	
0	satedner	0.561076	0.001192	0.450341	0.795850	0.930623	0.767605	0.894239	0.597057	0.125559	0.751090	0.278110	0.45
1	Rahul	0.254787	0.427195	0.969418	0.213716	0.912283	0.974008	0.314478	0.691099	0.603735	0.979347	0.785053	0.3
2	0.854784	0.793532	0.031116	0.471809	0.772358	0.447498	0.452217	0.644761	0.218107	0.125786	0.519473	0.229090	0.93
3	0.562486	0.790980	0.744740	0.238140	0.171643	0.136453	0.936933	0.931844	0.380153	0.264501	0.673073	0.232032	0.12
4	0.702867	0.270176	0.268086	0.076133	0.174052	0.712256	0.812254	0.648388	0.217212	0.763016	0.771966	0.055136	0.15
...	...	...	...	...	...	...	...	...	...	...	...	...	
3338	0.242589	0.189695	0.527391	0.300127	0.902239	0.026674	0.125397	0.918755	0.432655	0.011054	0.326422	0.905210	0.2
3339	0.887061	0.909481	0.001816	0.454096	0.124479	0.180765	0.284239	0.307691	0.909599	0.909509	0.890847	0.829765	0.06
3340	0.531965	0.708004	0.179359	0.905226	0.234289	0.676948	0.990071	0.162226	0.992508	0.565066	0.343011	0.244501	0.4
3341	0.060029	0.033379	0.814435	0.271410	0.507328	0.512107	0.554627	0.549610	0.743996	0.252561	0.118287	0.719574	0.23
3342	0.930778	0.048853	0.275265	0.415583	0.517176	0.010622	0.092440	0.338265	0.599631	0.229081	0.001652	0.961243	0.28

3343 rows × 20 columns

In [144…
```
newdf
```

Out[144…

	0	1	2	3	4	5	6	7	8	9	10	11	
0	satedner	0.561076	0.001192	0.450341	0.795850	0.930623	0.767605	0.894239	0.597057	0.125559	0.751090	0.278110	0.45
1	Rahul	0.254787	0.427195	0.969418	0.213716	0.912283	0.974008	0.314478	0.691099	0.603735	0.979347	0.785053	0.3
2	0.854784	0.793532	0.031116	0.471809	0.772358	0.447498	0.452217	0.644761	0.218107	0.125786	0.519473	0.229090	0.93
3	0.562486	0.790980	0.744740	0.238140	0.171643	0.136453	0.936933	0.931844	0.380153	0.264501	0.673073	0.232032	0.12
4	0.702867	0.270176	0.268086	0.076133	0.174052	0.712256	0.812254	0.648388	0.217212	0.763016	0.771966	0.055136	0.15
...	...	...	...	...	...	...	...	...	...	...	...	...	
3338	0.242589	0.189695	0.527391	0.300127	0.902239	0.026674	0.125397	0.918755	0.432655	0.011054	0.326422	0.905210	0.2
3339	0.887061	0.909481	0.001816	0.454096	0.124479	0.180765	0.284239	0.307691	0.909599	0.909509	0.890847	0.829765	0.06
3340	0.531965	0.708004	0.179359	0.905226	0.234289	0.676948	0.990071	0.162226	0.992508	0.565066	0.343011	0.244501	0.4
3341	0.060029	0.033379	0.814435	0.271410	0.507328	0.512107	0.554627	0.549610	0.743996	0.252561	0.118287	0.719574	0.23
3342	0.930778	0.048853	0.275265	0.415583	0.517176	0.010622	0.092440	0.338265	0.599631	0.229081	0.001652	0.961243	0.28

3343 rows × 20 columns

In [145…
```
ser = pd.Series(np.random.rand(34))
```

In [ ]:
```
newdf = pd.DataFrame(np.random.rand(3343,20), index=np.arange(3343))
```

In [146…
```
newdf
```

Out[146…

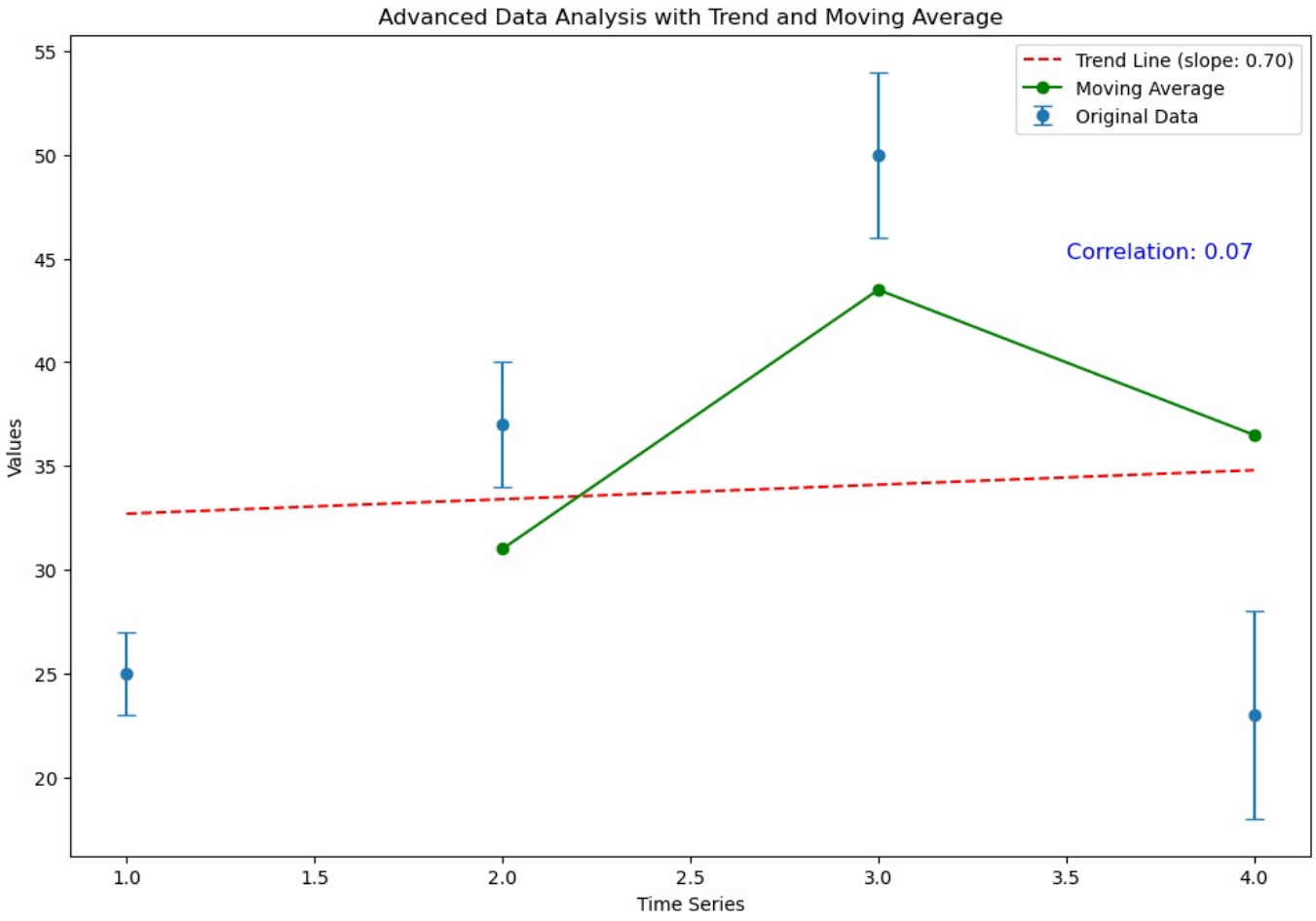
	0	1	2	3	4	5	6	7	8	9	10	11	
0	satedner	0.561076	0.001192	0.450341	0.795850	0.930623	0.767605	0.894239	0.597057	0.125559	0.751090	0.278110	0.45
1	Rahul	0.254787	0.427195	0.969418	0.213716	0.912283	0.974008	0.314478	0.691099	0.603735	0.979347	0.785053	0.3
2	0.854784	0.793532	0.031116	0.471809	0.772358	0.447498	0.452217	0.644761	0.218107	0.125786	0.519473	0.229090	0.93
3	0.562486	0.790980	0.744740	0.238140	0.171643	0.136453	0.936933	0.931844	0.380153	0.264501	0.673073	0.232032	0.12
4	0.702867	0.270176	0.268086	0.076133	0.174052	0.712256	0.812254	0.648388	0.217212	0.763016	0.771966	0.055136	0.15
...	...	...	...	...	...	...	...	...	...	...	...	...	...
3338	0.242589	0.189695	0.527391	0.300127	0.902239	0.026674	0.125397	0.918755	0.432655	0.011054	0.326422	0.905210	0.2
3339	0.887061	0.909481	0.001816	0.454096	0.124479	0.180765	0.284239	0.307691	0.909599	0.909509	0.890847	0.829765	0.06
3340	0.531965	0.708004	0.179359	0.905226	0.234289	0.676948	0.990071	0.162226	0.992508	0.565066	0.343011	0.244501	0.4
3341	0.060029	0.033379	0.814435	0.271410	0.507328	0.512107	0.554627	0.549610	0.743996	0.252561	0.118287	0.719574	0.23
3342	0.930778	0.048853	0.275265	0.415583	0.517176	0.010622	0.092440	0.338265	0.599631	0.229081	0.001652	0.961243	0.28

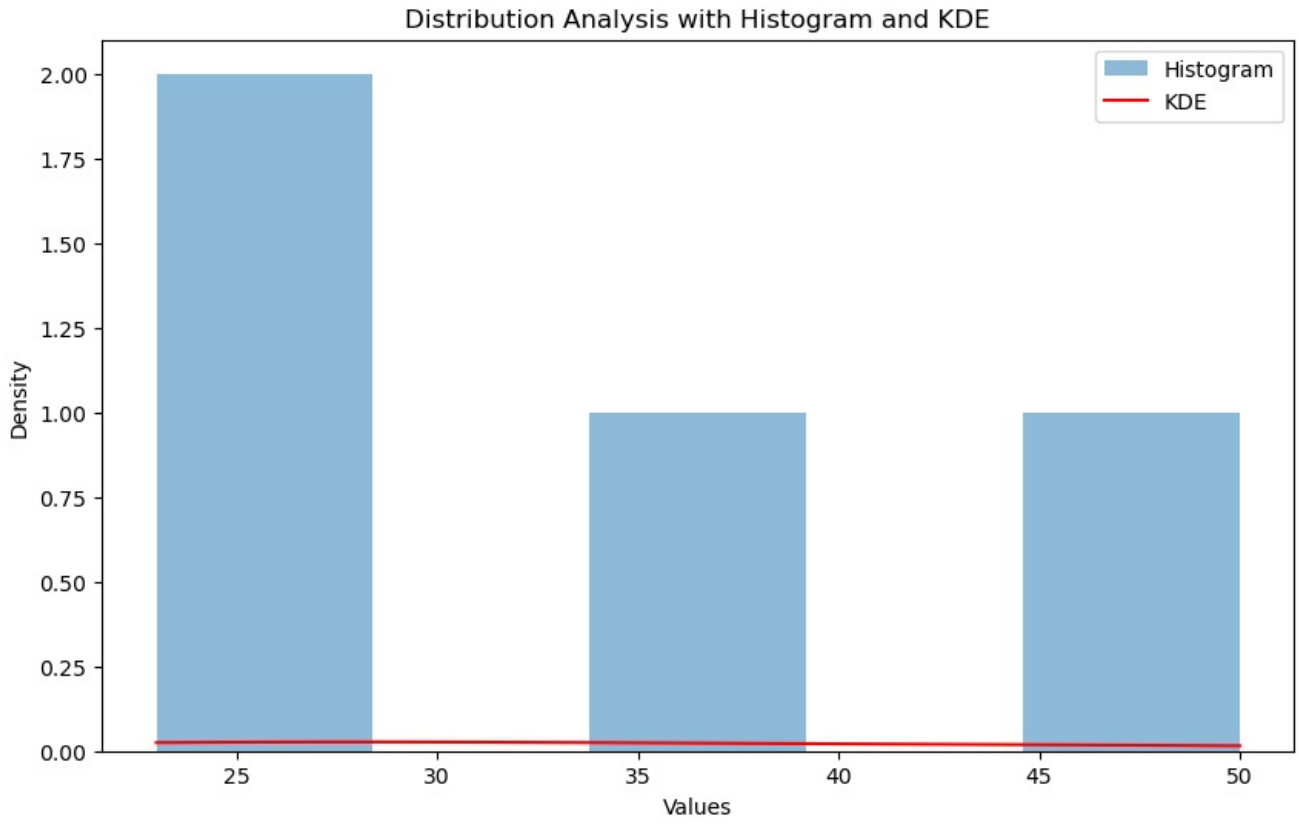
3343 rows × 20 columns

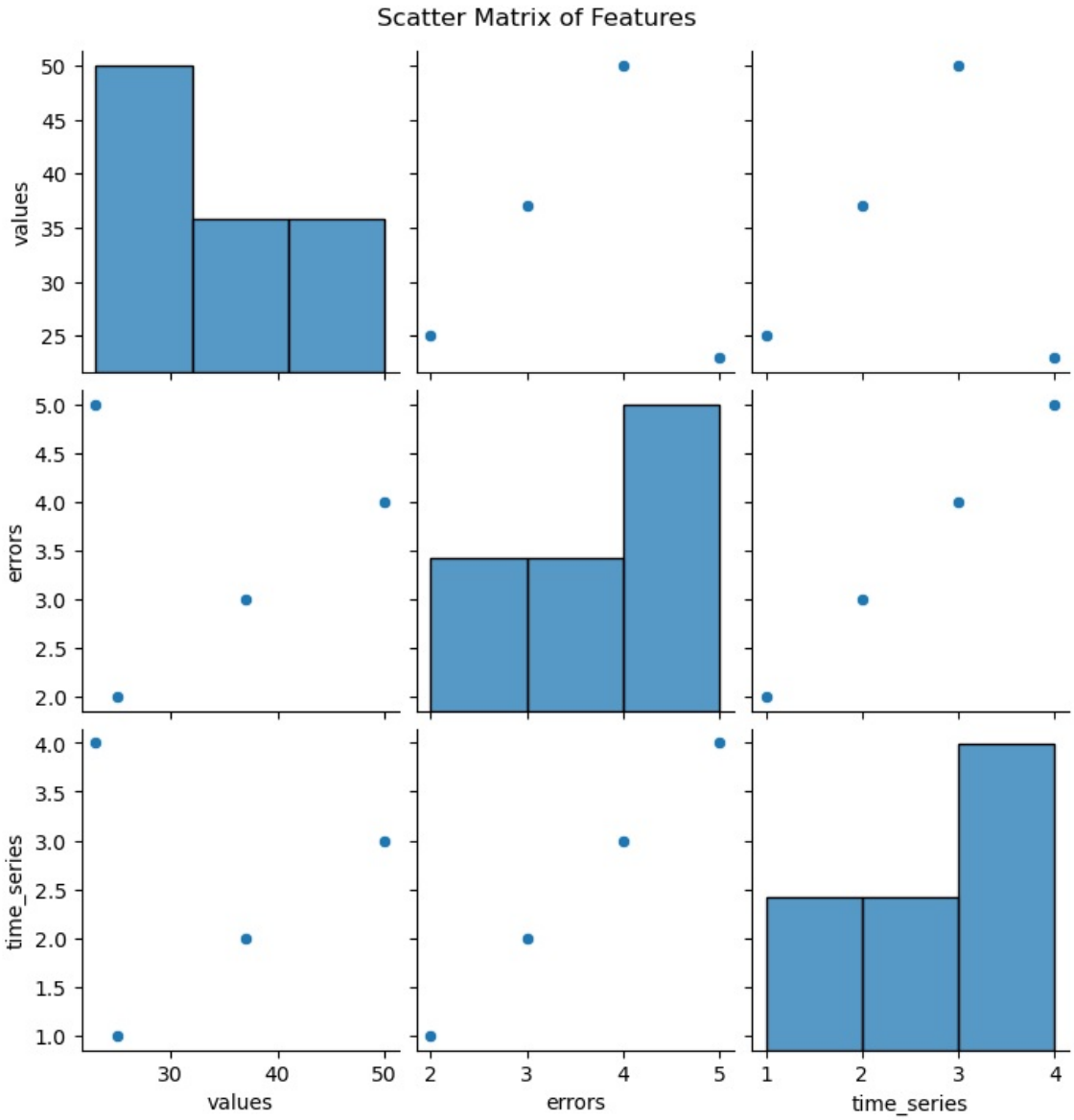
In [158…
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Sample Data Generation (Replace with your actual data loading)
# This time, let's assume we have time-series data
data = {
    'category': ['A', 'B', 'C', 'D'],
    'values': [25, 37, 50, 23],
    'errors': [2, 3, 4, 5],
    'time_series': [1, 2, 3, 4]
}

df = pd.DataFrame(data)

# Convert data to NumPy arrays for detailed analysis
values = np.array(df['values'])
errors = np.array(df['errors'])
time_series = np.array(df['time_series'])

# 1. Trend Analysis using Polyfit (Polynomial Fitting)
trend = np.polyfit(time_series, values, 1)  # Linear trend
trend_line = np.polyval(trend, time_series)

# 2. Correlation Calculation using NumPy
correlation = np.corrcoef(time_series, values)[0, 1]
print(f"Correlation between Time and Values: {correlation:.2f}")

# 3. Moving Average Calculation
window_size = 2
moving_avg = np.convolve(values, np.ones(window_size)/window_size, mode='valid')

# Visualization using Matplotlib

plt.figure(figsize=(12, 8))

# Original Data with Error Bars
plt.errorbar(df['time_series'], values, yerr=errors, fmt='o', label='Original Data', capsize=5)

# Trend Line
plt.plot(time_series, trend_line, label=f'Trend Line (slope: {trend[0]:.2f})', color='r', linestyle='--')

# Moving Average
plt.plot(time_series[window_size-1:], moving_avg, label='Moving Average', color='g', marker='o')

# Annotations for Correlation
plt.text(3.5, np.max(values) - 5, f'Correlation: {correlation:.2f}', fontsize=12, color='b')

# Adding titles and labels
plt.title('Advanced Data Analysis with Trend and Moving Average')
plt.xlabel('Time Series')
plt.ylabel('Values')
plt.legend()

# Show plot
```

```
plt.show()

# 4. Histogram and KDE Plot for Distribution Analysis
plt.figure(figsize=(10, 6))

# Histogram
plt.hist(values, bins=5, alpha=0.5, label='Histogram')

# Kernel Density Estimation (KDE) using Matplotlib
from scipy.stats import gaussian_kde
kde = gaussian_kde(values)
kde_x = np.linspace(min(values), max(values), 100)
kde_y = kde(kde_x)

plt.plot(kde_x, kde_y, color='r', label='KDE')

# Adding titles and labels
plt.title('Distribution Analysis with Histogram and KDE')
plt.xlabel('Values')
plt.ylabel('Density')
plt.legend()

# Show plot
plt.show()

# 5. Scatter Matrix for Pairwise Relationships
import seaborn as sns

# Assuming the dataframe has multiple numerical columns
sns.pairplot(df)
plt.suptitle('Scatter Matrix of Features', y=1.02)

plt.show()
```

```
Correlation between Time and Values: 0.07
```




Distribution Analysis with Histogram and KDE
Histogram
KDE
Density
Values
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
25
30
35
40
45
50



In [162…
```
newdf.describe()
```

Out[162…

	1	2	3	4	5	6	7	8	9	
count	3343.000000	3343.000000	3343.000000	3343.000000	3343.000000	3343.000000	3343.000000	3343.000000	3343.000000	334
mean	0.491004	0.491591	0.501394	0.493043	0.505059	0.505016	0.503800	0.505401	0.503050	
std	0.288965	0.287712	0.286441	0.289059	0.291096	0.290702	0.291508	0.283214	0.287236	
min	0.000288	0.000137	0.000232	0.000127	0.000114	0.000035	0.000412	0.000371	0.000129	
25%	0.232902	0.241878	0.255746	0.245966	0.249273	0.253529	0.252640	0.265151	0.252090	
50%	0.490917	0.487950	0.502484	0.485434	0.510141	0.511370	0.509279	0.505908	0.511891	
75%	0.742270	0.741858	0.748114	0.742720	0.754881	0.763551	0.760554	0.745003	0.749311	
max	0.999383	0.999779	0.999549	0.999999	0.999499	0.999996	0.999889	0.999710	0.999620	

In [164…
```
newdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3343 entries, 0 to 3342
Data columns (total 20 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   0       3343 non-null   object 
 1   1       3343 non-null   float64
 2   2       3343 non-null   float64
 3   3       3343 non-null   float64
 4   4       3343 non-null   float64
 5   5       3343 non-null   float64
 6   6       3343 non-null   float64
 7   7       3343 non-null   float64
 8   8       3343 non-null   float64
 9   9       3343 non-null   float64
 10  10      3343 non-null   float64
 11  11      3343 non-null   float64
 12  12      3343 non-null   float64
 13  13      3343 non-null   float64
 14  14      3343 non-null   float64
 15  15      3343 non-null   float64
 16  16      3343 non-null   float64
 17  17      3343 non-null   float64
 18  18      3343 non-null   float64
 19  19      3343 non-null   float64
dtypes: float64(19), object(1)
memory usage: 632.4+ KB
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Pandas, Numpy, Matplotlib, seaborn and more

In [39]:

```
# Importing necessary libraries for data analysis
import pandas as pd  # For data manipulation and analysis
import numpy as np  # For numerical computations
import matplotlib.pyplot as plt  # For creating static, animated, and int
eractive visualizations
import seaborn as sns  # For statistical data visualization based on Matp
lotlib
import scipy  # For scientific and technical computing (including optimiz
ation, integration, and statistics)
from sklearn.preprocessing import StandardScaler, LabelEncoder  # For pre
processing data (scaling, encoding)
from sklearn.model_selection import train_test_split  # For splitting dat
a into training and testing sets
from sklearn.linear_model import LinearRegression  # For linear regressio
n models
from sklearn.metrics import mean_squared_error, r2_score  # For model eva
luation metrics
import statsmodels.api as sm  # For statistical modeling and hypothesis t
esting
```

In [40]:

```
dict1 = {
    "name":['satender', 'rohan', 'rahul', 'ajay'],
    "marks":[92, 34, 34, 17],
    "city":['london', 'ontario', 'ajex', 'windsor']
}
```

In [41]:

```
df = pd.DataFrame(dict1)
```

In [42]:

```
df
```

Out[42]:

	name	marks	city
0	satender	92	london
1	rohan	34	ontario
2	rahul	34	ajex
3	ajay	17	windsor

In [43]:

```
import pandas as pd
import numpy as np
```

```
dict1 = {
    "name": ['satender', 'rohan', 'rahul', 'ajay'],
    "marks": [92, 34, 34, 17],
    "city": ['london', 'ontario', 'ajex', 'windsor']
}
```

In [44]:

```
df = pd.DataFrame(dict1)
```

In [45]:

```
df
```

Out[45]:

	name	marks	city
0	satender	92	london
1	rohan	34	ontario
2	rahul	34	ajex
3	ajay	17	windsor

In [46]:

```
import pandas as pd
import numpy as np

# Define the DataFrame
dict1 = {
    "name": ['satender', 'rohan', 'rahul', 'ajay'],
    "marks": [92, 34, 34, 17],
    "city": ['london', 'ontario', 'ajex', 'windsor']
}

df = pd.DataFrame(dict1)

# Add a calculated column 'grade' based on marks
df['grade'] = np.select(
    [
        df['marks'] >= 90,
        df['marks'] >= 75,
        df['marks'] >= 50,
        df['marks'] >= 35
    ],
    ['A', 'B', 'C', 'D'],
    default='F'
)
```

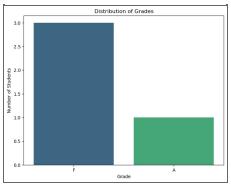
In [47]:

```
print(df)
```

```
       name  marks     city grade
  0  satender     92   london     A
  1     rohan     34  ontario     F
  2     rahul     34     ajex     F
  3      ajay     17  windsor     F
```

In [48]:

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

# Count the number of students in each grade
grade_counts = df['grade'].value_counts()

# Create a bar plot with the updated syntax
plt.figure(figsize=(8, 6))
sns.barplot(x=grade_counts.index, y=grade_counts.values, hue=grade_counts
.index, dodge=False, palette='viridis', legend=False)
plt.title('Distribution of Grades')
plt.xlabel('Grade')
plt.ylabel('Number of Students')
plt.show()
```



In [49]:

```
import pandas as pd
import numpy as np

# Original DataFrame
dict1 = {
    "name": ['satender', 'rohan', 'rahul', 'ajay'],
    "marks": [92, 34, 34, 17],
    "city": ['london', 'ontario', 'ajex', 'windsor']
}
df = pd.DataFrame(dict1)

# Add a column for pass/fail status based on marks
df['status'] = df['marks'].apply(lambda x: 'Pass' if x >= 35 else 'Fail')

# Add a column for a custom grade based on multiple criteria
def custom_grade(row):
    if row['marks'] >= 90:
        return 'A+'
    elif row['marks'] >= 75:
        return 'A'
    elif row['marks'] >= 50:
        return 'B'
    elif row['marks'] >= 35:
        return 'C'
    else:
        return 'F'

df['custom_grade'] = df.apply(custom_grade, axis=1)

print(df)
```

```
      name  marks     city status custom_grade
0  satender     92   london   Pass           A+
1     rohan     34  ontario   Fail            F
2     rahul     34     ajex   Fail            F
3      ajay     17  windsor   Fail            F
```

In [50]:

```
# Group by city and calculate the mean, sum, and count of marks
city_group = df.groupby('city')['marks'].agg(['mean', 'sum', 'count'])

# Add a column for the rank of each student within their city
df['rank_within_city'] = df.groupby('city')['marks'].rank(ascending=False
)

print(city_group)
print(df)
```

```
          mean  sum  count
city
ajex      34.0   34      1
london    92.0   92      1
ontario   34.0   34      1
windsor   17.0   17      1
       name  marks     city status custom_grade  rank_within_cit
y
0  satender     92   london   Pass           A+               1.
0
1     rohan     34  ontario   Fail            F               1.
0
2     rahul     34     ajex   Fail            F               1.
0
3      ajay     17  windsor   Fail            F               1.
0
```

In [51]:

```
# Create a pivot table to show the average marks and count of students by
 city and status
pivot_table = pd.pivot_table(df, values='marks', index='city', columns='s
tatus', aggfunc={'marks': ['mean', 'count']})

print(pivot_table)
```

```
         count       mean
status    Fail Pass  Fail  Pass
city
ajex       1.0  NaN  34.0   NaN
london     NaN  1.0   NaN  92.0
ontario    1.0  NaN  34.0   NaN
windsor    1.0  NaN  17.0   NaN
```

In [52]:

```
# Additional DataFrame with more student information
additional_info = pd.DataFrame({
    'name': ['satender', 'rohan', 'rahul', 'ajay', 'vikram'],
    'age': [22, 21, 22, 23, 24],
    'gender': ['M', 'M', 'M', 'M', 'M']
})

# Merge on the 'name' column
merged_df = pd.merge(df, additional_info, on='name', how='left')

print(merged_df)
```

```
        name  marks     city status custom_grade  rank_within_cit
y  age gender
0  satender     92   london   Pass           A+               1.
0   22      M
1     rohan     34  ontario   Fail            F               1.
0   21      M
2     rahul     34     ajex   Fail            F               1.
0   22      M
3      ajay     17  windsor   Fail            F               1.
0   23      M
```

In [53]:

```
# Introduce some missing data for demonstration
merged_df.loc[merged_df['name'] == 'vikram', 'marks'] = np.nan

# Fill missing data with the mean of the column
merged_df['marks_filled'] = merged_df['marks'].fillna(merged_df['marks'].
mean())

# Alternatively, interpolate missing values based on surrounding data
merged_df['marks_interpolated'] = merged_df['marks'].interpolate()

print(merged_df)
```

```
        name  marks     city status custom_grade  rank_within_cit
y  age gender  \
0  satender   92.0   london   Pass           A+               1.
0   22      M
1     rohan   34.0  ontario   Fail            F               1.
0   21      M
2     rahul   34.0     ajex   Fail            F               1.
0   22      M
3      ajay   17.0  windsor   Fail            F               1.
0   23      M

   marks_filled  marks_interpolated
0          92.0                92.0
1          34.0                34.0
2          34.0                34.0
3          17.0                17.0
```

In [54]:

```
# Filter students who passed and scored more than the average marks of th
e class
average_marks = df['marks'].mean()
filtered_students = df[(df['status'] == 'Pass') & (df['marks'] > average_
marks)]

print(filtered_students)
```

```
        name  marks    city status custom_grade  rank_within_city
0  satender     92  london   Pass           A+               1.0
```

In [55]:

```
# Calculate a rolling average of marks with a window size of 2
df['rolling_avg'] = df['marks'].rolling(window=2).mean()

print(df)
```

```
      name  marks     city status custom_grade  rank_within_cit
y  rolling_avg
0  satender     92   london   Pass           A+               1.
0         NaN
1     rohan     34  ontario   Fail            F               1.
0        63.0
2     rahul     34     ajex   Fail            F               1.
0        34.0
3      ajay     17  windsor   Fail            F               1.
0        25.5
```

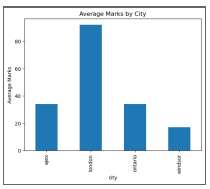
In [56]:

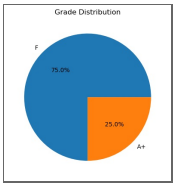
```
# Custom function to generate a summary for each row
def summarize_row(row):
    return f"{row['name']} from {row['city']} scored {row['marks']} and g
ot grade {row['custom_grade']}."

df['summary'] = df.apply(summarize_row, axis=1)

print(df['summary'])
```

```
0    satender from london scored 92 and got grade A+.
1       rohan from ontario scored 34 and got grade F.
2          rahul from ajex scored 34 and got grade F.
3        ajay from windsor scored 17 and got grade F.
Name: summary, dtype: object
```

In [57]:

```
import matplotlib.pyplot as plt

# Bar plot for average marks by city
df.groupby('city')['marks'].mean().plot(kind='bar', title='Average Marks
by City')
plt.ylabel('Average Marks')
plt.show()

# Pie chart for distribution of grades
df['custom_grade'].value_counts().plot(kind='pie', title='Grade Distribut
ion', autopct='%1.1f%%')
plt.ylabel('')
plt.show()
```





In [58]:

```
# Set a MultiIndex with 'city' and 'name'
df_multiindex = df.set_index(['city', 'name'])
```

```
# Access data using the MultiIndex
print(df_multiindex.loc['london'].loc['satender'])

# Reset the index to default
df_reset = df_multiindex.reset_index()
print(df_reset)
```

```
marks
92
status
Pass
custom_grade
A+
rank_within_city
1.0
rolling_avg
NaN
summary             satender from london scored 92 and got grade
A+.
Name: satender, dtype: object
      city      name  marks status custom_grade  rank_within_cit
y  \
0   london  satender     92   Pass           A+               1.
0
1  ontario     rohan     34   Fail            F               1.
0
2     ajex     rahul     34   Fail            F               1.
0
3  windsor      ajay     17   Fail            F               1.
0

   rolling_avg                                              summary
0          NaN  satender from london scored 92 and got grade A+.
1         63.0     rohan from ontario scored 34 and got grade F.
2         34.0        rahul from ajex scored 34 and got grade F.
3         25.5      ajay from windsor scored 17 and got grade F.
```

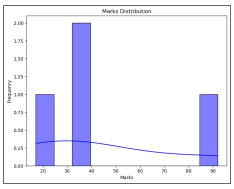
In [59]:

```
# Convert 'custom_grade' to a categorical type with order
grades = ['F', 'D', 'C', 'B', 'A', 'A+']
df['custom_grade'] = pd.Categorical(df['custom_grade'], categories=grades
, ordered=True)

# Sort DataFrame by categorical grade
df_sorted_by_grade = df.sort_values('custom_grade')

print(df_sorted_by_grade)
```

```
        name  marks     city status custom_grade  rank_within_cit
y  \
1     rohan     34  ontario   Fail            F               1.
0
2     rahul     34     ajex   Fail            F               1.
0
3      ajay     17  windsor   Fail            F               1.
0
0  satender     92   london   Pass           A+               1.
0

   rolling_avg                                              summary
1         63.0     rohan from ontario scored 34 and got grade F.
2         34.0        rahul from ajex scored 34 and got grade F.
3         25.5      ajay from windsor scored 17 and got grade F.
0          NaN  satender from london scored 92 and got grade A+.
```

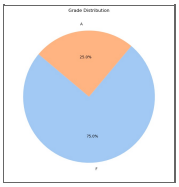
In [60]:

```
plt.figure(figsize=(8, 6))
sns.histplot(df['marks'], bins=10, kde=True, color='blue')
plt.title('Marks Distribution')
plt.xlabel('Marks')
plt.ylabel('Frequency')
plt.show()
```



In [61]:

```
plt.figure(figsize=(8, 8))
plt.pie(grade_counts, labels=grade_counts.index, autopct='%1.1f%%', start
angle=140, colors=sns.color_palette('pastel'))
plt.title('Grade Distribution')
plt.show()
```



In [66]:

```
# Binning an array of data
data = np.random.randn(1000)  # Generate 1000 random numbers
bins = np.linspace(-3, 3, 7)  # Create 6 bins between -3 and 3
digitized = np.digitize(data, bins)

# Counting the number of elements in each bin
bin_counts = np.bincount(digitized)
print("Bin Counts:", bin_counts)
```

```
Bin Counts: [  1  26 135 334 346 131  26   1]
```

In [67]:

```
import numpy as np

# Vectorized operation using ufunc
arr = np.array([1, 2, 3, 4, 5])

# Example of vectorized operation
result = np.exp(arr)  # Exponential function applied element-wise
print("Exponential of Array:", result)

# Custom ufunc
def custom_func(x, y):
    return x + y * y

# Vectorized using np.vectorize
vectorized_func = np.vectorize(custom_func)
result = vectorized_func(arr, arr)
print("Custom Ufunc Result:", result)
```

```
Exponential of Array: [  2.71828183   7.3890561   20.08553692  5
4.59815003 148.4131591 ]
Custom Ufunc Result: [ 2  6 12 20 30]
```

In [68]:

```
# Example of broadcasting without memory duplication
arr1 = np.array([[1], [2], [3]])
arr2 = np.array([4, 5, 6])

# Broadcasting adds arr2 to each row of arr1 without copying data
result = arr1 + arr2
print("Broadcasted Result:\n", result)

# Memory-efficient operation
print("Memory Address of arr1:", arr1.__array_interface__['data'])
print("Memory Address of arr2:", arr2.__array_interface__['data'])
```

```
Broadcasted Result:
 [[5 6 7]
 [6 7 8]
 [7 8 9]]
Memory Address of arr1: (2162781248528, False)
Memory Address of arr2: (2162781248048, False)
```

In [69]:

```
# Integer array indexing
arr = np.array([[1, 2], [3, 4], [5, 6]])
print("Original Array:\n", arr)

# Select elements using an integer array
result = arr[[0, 1, 2], [1, 0, 1]]  # Selects (0,1), (1,0), and (2,1)
print("Integer Array Indexing Result:", result)

# Boolean indexing
mask = arr > 2
print("Boolean Mask:\n", mask)
result = arr[mask]
print("Boolean Indexing Result:", result)
```

```
Original Array:
 [[1 2]
 [3 4]
 [5 6]]
Integer Array Indexing Result: [2 3 6]
Boolean Mask:
 [[False False]
 [ True  True]
 [ True  True]]
Boolean Indexing Result: [3 4 5 6]
```

In [70]:

```python
# Create a structured array with multiple data types
data = np.array([
    ('Satender', 92, 'London'),
    ('Rohan', 34, 'Ontario'),
    ('Rahul', 34, 'Ajex'),
    ('Ajay', 17, 'Windsor')
], dtype=[('name', 'U10'), ('marks', 'i4'), ('city', 'U10')])

print("Structured Array:\n", data)

# Accessing specific fields
print("Names:", data['name'])
print("Marks:", data['marks'])
```

```
Structured Array:
 [('Satender', 92, 'London') ('Rohan', 34, 'Ontario') ('Rahul',
34, 'Ajex')
 ('Ajay', 17, 'Windsor')]
Names: ['Satender' 'Rohan' 'Rahul' 'Ajay']
Marks: [92 34 34 17]
```

In [71]:

```python
# Create a masked array
arr = np.array([1, 2, 3, -1, 5])
masked_arr = np.ma.masked_array(arr, mask=[0, 0, 0, 1, 0])

print("Masked Array:", masked_arr)
print("Mean ignoring masked elements:", np.ma.mean(masked_arr))
```

```
Masked Array: [1 2 3 -- 5]
Mean ignoring masked elements: 2.75
```

In [72]:

```python
# Fancy indexing with arrays of indices
arr = np.arange(10)
print("Original Array:", arr)

# Fancy indexing to reorder elements
indices = [3, 1, 9, 7]
result = arr[indices]
print("Fancy Indexing Result:", result)

# Multi-indexing with slices
multi_indexed_result = arr[::2]  # Access every second element
print("Multi-Indexing Result:", multi_indexed_result)
```

```
Original Array: [0 1 2 3 4 5 6 7 8 9]
Fancy Indexing Result: [3 1 9 7]
Multi-Indexing Result: [0 2 4 6 8]
```

In [73]:

```python
# Stacking arrays vertically and horizontally
arr1 = np.array([[1, 2], [3, 4]])
arr2 = np.array([[5, 6], [7, 8]])

# Vertical stacking
vstacked = np.vstack((arr1, arr2))
print("Vertically Stacked Arrays:\n", vstacked)

# Horizontal stacking
hstacked = np.hstack((arr1, arr2))
print("Horizontally Stacked Arrays:\n", hstacked)

# Splitting arrays
split_arr = np.hsplit(hstacked, 2)
print("Horizontally Split Arrays:", split_arr)
```

```
Vertically Stacked Arrays:
 [[1 2]
 [3 4]
 [5 6]
 [7 8]]
Horizontally Stacked Arrays:
 [[1 2 5 6]
 [3 4 7 8]]
Horizontally Split Arrays: [array([[1, 2],
       [3, 4]]), array([[5, 6],
       [7, 8]])]
```

In [74]:

```python
from numpy.linalg import inv, eig, solve

# Define a matrix
matrix = np.array([[1, 2], [3, 4]])

# Inverse of a matrix
matrix_inv = inv(matrix)
print("Inverse of Matrix:\n", matrix_inv)

# Eigenvalues and eigenvectors
eigenvalues, eigenvectors = eig(matrix)
print("Eigenvalues:", eigenvalues)
print("Eigenvectors:\n", eigenvectors)

# Solving a system of linear equations
# For example, solving Ax = b, where A is the matrix, and b is a vector
A = np.array([[3, 1], [1, 2]])
b = np.array([9, 8])
x = solve(A, b)
print("Solution of Linear Equations:", x)
```

```
Inverse of Matrix:
 [[-2.   1. ]
 [ 1.5 -0.5]]
Eigenvalues: [-0.37228132  5.37228132]
Eigenvectors:
 [[-0.82456484 -0.41597356]
 [ 0.56576746 -0.90937671]]
Solution of Linear Equations: [2. 3.]
```

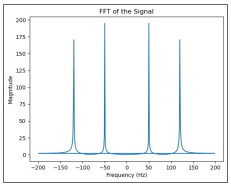
In [75]:

```python
# Binning an array of data
data = np.random.randn(1000)  # Generate 1000 random numbers
bins = np.linspace(-3, 3, 7)  # Create 6 bins between -3 and 3
digitized = np.digitize(data, bins)

# Counting the number of elements in each bin
bin_counts = np.bincount(digitized)
print("Bin Counts:", bin_counts)
```

```
Bin Counts: [  1  26 151 307 343 149  23]
```

In [77]:

```python
# Generate a sample signal
time = np.linspace(0, 1, 400)
signal = np.sin(2 * np.pi * 50 * time) + np.sin(2 * np.pi * 120 * time)

# Perform FFT
signal_fft = np.fft.fft(signal)
frequencies = np.fft.fftfreq(len(signal), d=time[1] - time[0])

# Plot the FFT result
import matplotlib.pyplot as plt
plt.plot(frequencies, np.abs(signal_fft))
plt.title('FFT of the Signal')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Magnitude')
plt.show()
```



In [78]:

```python
from scipy import optimize

# Define a simple quadratic function
def f(x):
    return x**2 + 10*np.sin(x)

# Find the minimum of the function
result = optimize.minimize(f, x0=0)  # Start the search at x=0
print("Function minimum:", result.x)

# Numerical integration
from scipy.integrate import quad
```

```
result, error = quad(np.sin, 0, np.pi)
print("Integral of sin(x) from 0 to pi:", result)
```

```
Function minimum: [-1.30644012]
Integral of sin(x) from 0 to pi: 2.0
```

In [79]:

```
# Normalizing data
data = np.random.rand(100, 3)  # Random dataset with 3 features
normalized_data = (data - np.mean(data, axis=0)) / np.std(data, axis=0)
print("Normalized Data:\n", normalized_data)

# Generating polynomial features
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=2)
poly_features = poly.fit_transform(data)
print("Polynomial Features:\n", poly_features)
```

```
Normalized Data:
 [[-0.07867606  1.54618142 -1.29728695]
 [-0.10420612 -1.27499644  0.53398214]
 [ 1.50242555 -0.28922969 -0.06329412]
 [-1.11531425 -1.24799388 -0.51755026]
 [ 1.35492871 -1.26585454  0.44063684]
 [ 1.35306439 -1.06624746  0.48733713]
 [-1.55077168 -1.39048182  1.92834303]
 [-0.05628639 -0.86397572  0.6636992 ]
 [-0.51667831  0.7977934   1.76684069]
 [ 0.99008777 -0.06148516 -0.26800809]
 [-0.46113287 -1.23992938 -0.25505175]
 [-0.95954609  0.11906083  0.67255312]
 [-1.59912787  0.06741177 -1.06307189]
 [-0.97347682  0.57885872  0.51008799]
 [ 0.25459418  0.16654153 -1.58046817]
 [ 0.5041768  -0.29322821  0.82043648]
 [-0.6093574   0.63011243 -0.66525166]
 [-0.18071358  0.40110395  1.64799404]
 [-0.43537875  0.33215211  0.42583036]
 [-0.92600611  0.21720443 -1.45598939]
 [-0.05346763 -0.42508215  1.49828225]
 [ 0.91533587  1.28506151  1.02704189]
 [-0.55417338 -0.53244166 -1.23664048]
 [-0.44137591  1.14908844 -0.07498698]
 [ 1.04614549  0.74272252  0.80511854]
 [-1.28817554  1.41955168 -0.47810675]
 [-1.00221074  1.40352547  0.89215949]
 [ 0.07416694 -1.5301512  -1.07390482]
 [-0.51508739  0.223562   -0.09120319]
 [-1.24062298  0.46283781  0.78932898]
 [-0.31027573  1.33046387 -0.79222998]
 [-0.56744718 -0.54464557  0.85576963]
 [ 0.82079411 -0.93264093  0.83734565]
 [ 0.09029362  0.06095115  1.35513143]
 [-1.11100974 -0.98067538 -0.76332469]
 [-1.06204529 -1.40980879  0.79301311]
 [ 0.90490182  1.16839234  1.41626827]
 [-1.67586776  0.58698027 -1.64810253]
 [ 1.27820901  1.17631363  1.86175951]
 [ 1.07748639 -1.27403474  0.36061581]
 [ 0.68940796  0.94352287 -0.54891891]
```

```
 [-0.9583129   0.94053006 -0.24575884]
 [ 1.28847543  1.65583783 -0.55437477]
 [-0.73414333  1.62686212  1.01402331]
 [ 1.44931944 -0.88037475 -0.93627462]
 [-0.61603356 -1.38947094 -1.29045207]
 [ 0.95026891 -0.69322642 -0.06243894]
 [ 1.00933614 -0.75955669 -1.15800272]
 [-0.61682381  0.46518041 -1.17582128]
 [-0.92588471  1.01377516  0.32366054]
 [-1.23839256  1.38691132  0.01745632]
 [ 1.50539223 -1.20861454  0.26728214]
 [ 1.4532346   1.65117695  1.61900853]
 [-1.44980081  0.44703516 -0.69439922]
 [-0.53740531  0.75766779  1.32930537]
 [ 0.27272659 -0.82622964  0.31405507]
 [-1.62728456  1.23041443 -0.16932265]
 [ 0.29366117  1.266253    1.65489094]
 [ 0.88696677 -0.81814617  1.08221023]
 [ 1.43176692  0.31444075 -1.55874555]
 [-1.11111929 -0.25382396  0.78451901]
 [-0.50978095  0.21930307  1.24785109]
 [ 0.76290696 -0.83431869 -0.03752822]
 [ 1.55685624  1.09443153 -1.44459393]
 [ 1.39221015 -1.37237802 -0.22068847]
 [ 1.27823625 -1.36676266  0.48330412]
 [-1.10990268 -0.017659    0.40519042]
 [-0.12278711 -1.15943009 -1.00601121]
 [-1.08304343 -0.73454301  0.37557261]
 [ 1.16121232  0.91959139 -1.62398821]
 [ 1.01761182 -1.3475367   1.46096009]
 [ 1.02449053 -0.88149665  0.6838838 ]
 [ 0.60425513 -0.98284355 -0.9986486 ]
 [ 0.77968183  1.00896766  0.23696891]
 [ 0.49948879 -1.27689378 -1.2558867 ]
 [-1.50526958 -0.12582776  0.41495445]
 [ 0.24423009 -1.38867114 -0.87197721]
 [ 1.25160994 -1.54205501 -0.05719881]
 [ 0.74645546  0.09373551  0.63144037]
 [ 1.4907586   0.23038493  0.33225524]
 [-0.38264888 -0.2689569  -0.40806976]
 [-1.49229964 -0.7215264  -1.57764736]
 [ 0.37020804  1.32955131 -1.19011621]
 [-0.39144937  0.95317271 -0.87230193]
 [-1.24783222  1.16285276 -0.61707654]
 [-1.0089783   0.14285115 -1.65867203]
 [-1.70312002  1.16327032 -1.45238822]
 [-0.92027631 -0.55061517 -0.10209376]
 [ 1.48219261  0.93961219  0.73998518]
 [-1.02163131 -1.39762379 -0.14760248]
 [-0.07641411 -1.45850756  0.71743082]
 [-0.63483676  0.14295218 -1.12512989]
 [ 0.75607758  0.75011468  1.61658544]
 [ 1.346191   -1.27263586 -0.13500214]
 [-0.04575234 -1.25811399 -1.45197508]
 [ 1.11362445 -0.16836523  1.81270239]
 [ 1.24647004  1.52636267 -1.14949633]
 [-1.59595063  1.58891332 -0.75772182]
 [ 0.74170811  1.23010978  0.88585877]
 [-0.20803671 -0.48254948 -0.96013454]]
Polynomial Features:
 [[1.00000000e+00 4.93365879e-01 9.64702515e-01 1.32791873e-01
  2.43409891e-01 4.75951305e-01 6.55149790e-02 9.30650942e-01
  1.28104653e-01 1.76336814e-02]
 [1.00000000e+00 4.85755180e-01 1.01855633e-01 6.21718442e-01
```

 2.35958095e-01 4.94769013e-02 3.02002954e-01 1.03745699e-02
 6.33255254e-02 3.86533821e-01]
[1.00000000e+00 9.64703865e-01 4.03348771e-01 4.62252946e-01
 9.30653548e-01 3.89112119e-01 4.45937204e-01 1.62690231e-01
 1.86449158e-01 2.13677786e-01]
[1.00000000e+00 1.84336429e-01 1.10114266e-01 3.40972084e-01
 3.39799191e-02 2.02980706e-02 6.28535764e-02 1.21251516e-02
 3.75458907e-02 1.16261962e-01]
[1.00000000e+00 9.20733978e-01 1.04651650e-01 5.96796382e-01
 8.47751058e-01 9.63563301e-02 5.49490706e-01 1.09519679e-02
 6.24557261e-02 3.56165921e-01]
[1.00000000e+00 9.20178211e-01 1.65700739e-01 6.09264791e-01
 8.46727939e-01 1.52474210e-01 5.60632186e-01 2.74567349e-02
 1.00955626e-01 3.71203586e-01]
[1.00000000e+00 5.45233757e-02 6.65348519e-02 9.93995836e-01
 2.97279850e-03 3.62770473e-03 5.41960084e-02 4.42688651e-03
 6.61353657e-02 9.88027721e-01]
[1.00000000e+00 5.00040404e-01 2.27564809e-01 6.56351319e-01
 2.50040406e-01 1.13791599e-01 3.28202179e-01 5.17857423e-02
 1.49362462e-01 4.30797054e-01]
[1.00000000e+00 3.62794196e-01 7.35810786e-01 9.50876677e-01
 1.31619629e-01 2.66947883e-01 3.44972540e-01 5.41417514e-01
 6.99665316e-01 9.04166455e-01]
[1.00000000e+00 8.11972218e-01 4.73003598e-01 4.07596806e-01
 6.59298882e-01 3.84065780e-01 3.30957283e-01 2.23732403e-01
 1.92794756e-01 1.66135157e-01]
[1.00000000e+00 3.79352697e-01 1.12580763e-01 4.11055991e-01
 1.43908469e-01 4.27078160e-02 1.55935199e-01 1.26744281e-02
 4.62769970e-02 1.68967028e-01]
[1.00000000e+00 2.30772061e-01 5.28222925e-01 6.58715207e-01
 5.32557441e-02 1.21899093e-01 1.52013066e-01 2.79019459e-01
 3.47948474e-01 4.33905724e-01]
[1.00000000e+00 4.01080421e-02 5.12426251e-01 1.95324445e-01
 1.60865504e-03 2.05524136e-02 7.83408108e-03 2.62580662e-01
 1.00089373e-01 3.81516390e-02]
[1.00000000e+00 2.26619208e-01 6.68850421e-01 6.15338994e-01
 5.13562653e-02 1.51574353e-01 1.39447635e-01 4.47360886e-01
 4.11569745e-01 3.78642077e-01]
[1.00000000e+00 5.92716182e-01 5.42744722e-01 5.71859322e-02
 3.51312472e-01 3.21693579e-01 3.38950274e-02 2.94571833e-01
 3.10373628e-02 3.27023084e-03]
[1.00000000e+00 6.67118592e-01 4.02125838e-01 6.98198266e-01
 4.45047215e-01 2.68265623e-01 4.65781044e-01 1.61705189e-01
 2.80763563e-01 4.87480819e-01]
[1.00000000e+00 3.35165880e-01 6.84526178e-01 3.01537606e-01
 1.12336167e-01 2.29429819e-01 1.01065117e-01 4.68576088e-01
 2.06410385e-01 9.09249281e-02]
[1.00000000e+00 4.62947747e-01 6.14484776e-01 9.19146067e-01
 2.14320616e-01 2.84474343e-01 4.25516601e-01 3.77591540e-01
 5.64801265e-01 8.44829493e-01]
[1.00000000e+00 3.87030190e-01 5.93396110e-01 5.92843232e-01
 1.49792368e-01 2.29662209e-01 2.29448229e-01 3.52118944e-01
 3.51790868e-01 3.51463098e-01]
[1.00000000e+00 2.40770575e-01 5.58239786e-01 9.04202528e-02
 5.79704700e-02 1.34407714e-01 2.17705363e-02 3.11631659e-01
 5.04761826e-02 8.17582212e-03]
[1.00000000e+00 5.00880696e-01 3.61798796e-01 8.79174838e-01
 2.50881471e-01 1.81218032e-01 4.40361705e-01 1.30898368e-01
 3.18084398e-01 7.72948397e-01]
[1.00000000e+00 7.89688126e-01 8.84839950e-01 7.53359396e-01
 6.23607336e-01 6.98747602e-01 5.94918970e-01 7.82941737e-01
 6.66602491e-01 5.67550380e-01]
[1.00000000e+00 3.51616638e-01 3.28963283e-01 1.48983742e-01
 1.23634260e-01 1.15668964e-01 5.23851625e-02 1.08216842e-01

```
 4.90101809e-02 2.21961553e-02]
[1.00000000e+00 3.85242394e-01 8.43253084e-01 4.59131096e-01
 1.48411702e-01 3.24856837e-01 1.76876763e-01 7.11075764e-01
 3.87163712e-01 2.10801363e-01]
[1.00000000e+00 8.28683435e-01 7.18967560e-01 6.94108561e-01
 6.86716235e-01 5.95796507e-01 5.75196266e-01 5.16914352e-01
 4.99041538e-01 4.81786694e-01]
[1.00000000e+00 1.32805212e-01 9.25973272e-01 3.51503020e-01
 1.76372243e-02 1.22974076e-01 4.66814330e-02 8.57426500e-01
 3.25482401e-01 1.23554373e-01]
[1.00000000e+00 2.18053415e-01 9.21071716e-01 7.17347437e-01
 4.75472918e-02 2.00842833e-01 1.56420058e-01 8.48373106e-01
 6.60728435e-01 5.14587345e-01]
[1.00000000e+00 5.38929499e-01 2.38174864e-02 1.92432185e-01
 2.90445005e-01 1.28359460e-02 1.03707381e-01 5.67272660e-04
 4.58325096e-03 3.70301459e-02]
[1.00000000e+00 3.63268460e-01 5.60184225e-01 4.54801565e-01
 1.31963974e-01 2.03497260e-01 1.65215064e-01 3.13806366e-01
 2.54772662e-01 2.06844464e-01]
[1.00000000e+00 1.46980978e-01 6.33365851e-01 6.89892941e-01
 2.16034078e-02 9.30927321e-02 1.01401139e-01 4.01152301e-01
 4.36954630e-01 4.75952270e-01]
[1.00000000e+00 4.24324320e-01 8.98726093e-01 2.67635938e-01
 1.80051128e-01 3.81351338e-01 1.13564437e-01 8.07708590e-01
 2.40531401e-01 7.16289955e-02]
[1.00000000e+00 3.47659622e-01 3.25230760e-01 7.07631786e-01
 1.20867213e-01 1.13069603e-01 2.46014999e-01 1.05775048e-01
 2.30143624e-01 5.00742744e-01]
[1.00000000e+00 7.61504535e-01 2.06563808e-01 7.02712808e-01
 5.79889157e-01 1.57299276e-01 5.35118990e-01 4.26686067e-02
 1.45155033e-01 4.93805290e-01]
[1.00000000e+00 5.43736980e-01 5.10450293e-01 8.40955312e-01
 2.95649903e-01 2.77550701e-01 4.57258501e-01 2.60559502e-01
 4.29265885e-01 7.07205836e-01]
[1.00000000e+00 1.85619637e-01 1.91872646e-01 2.75353301e-01
 3.44546498e-02 3.56153311e-02 5.11109798e-02 3.68151124e-02
 5.28327665e-02 7.58194402e-02]
[1.00000000e+00 2.00216298e-01 6.06237696e-02 6.90876558e-01
 4.00865658e-02 1.21378667e-02 1.38324747e-01 3.67524144e-03
 4.18835413e-02 4.77310419e-01]
[1.00000000e+00 7.86577660e-01 8.49157113e-01 8.57278104e-01
 6.18704415e-01 6.67928015e-01 6.74315805e-01 7.21067802e-01
 7.27963800e-01 7.34925748e-01]
[1.00000000e+00 1.72313160e-02 6.71334367e-01 3.91283796e-02
 2.96918251e-04 1.15679746e-02 6.74233475e-04 4.50689832e-01
 2.62682260e-02 1.53103009e-03]
[1.00000000e+00 8.97863272e-01 8.51579808e-01 9.76218847e-01
 8.06158454e-01 7.64602233e-01 8.76511048e-01 7.25188170e-01
 8.31328259e-01 9.53003238e-01]
[1.00000000e+00 8.38026387e-01 1.02149764e-01 5.75431741e-01
 7.02288225e-01 8.56041977e-02 4.82226983e-01 1.04345743e-02
 5.87802165e-02 3.31121688e-01]
[1.00000000e+00 7.22337360e-01 7.80381612e-01 3.32597036e-01
 5.21771262e-01 5.63698793e-01 2.40247265e-01 6.08995460e-01
 2.59552611e-01 1.10620788e-01]
[1.00000000e+00 2.31139686e-01 7.79466271e-01 4.13537087e-01
 5.34255543e-02 1.80165589e-01 9.55848322e-02 6.07567667e-01
 3.22338211e-01 1.71012922e-01]
[1.00000000e+00 9.00923765e-01 9.98240524e-01 3.31140387e-01
 8.11663631e-01 8.99338611e-01 2.98332244e-01 9.96484143e-01
 3.30557753e-01 1.09653956e-01]
[1.00000000e+00 2.97966279e-01 9.89378408e-01 7.49883593e-01
 8.87839032e-02 2.94801402e-01 2.23440023e-01 9.78869633e-01
 7.41918635e-01 5.62325403e-01]
```

```
[1.00000000e+00 9.48872545e-01 2.22549225e-01 2.29177774e-01
 9.00359107e-01 2.11170849e-01 2.17460498e-01 4.95281573e-02
 5.10033359e-02 5.25224521e-02]
[1.00000000e+00 3.33175667e-01 6.68440285e-02 1.34616702e-01
 1.11006025e-01 2.22708038e-02 4.48510095e-02 4.46812414e-03
 8.99832267e-03 1.81216565e-02]
[1.00000000e+00 8.00101923e-01 2.79787855e-01 4.62481270e-01
 6.40163087e-01 2.23858801e-01 3.70032153e-01 7.82812438e-02
 1.29396643e-01 2.13888925e-01]
[1.00000000e+00 8.17710295e-01 2.59500985e-01 1.69979066e-01
 6.68650127e-01 2.12196627e-01 1.38993633e-01 6.73407610e-02
 4.41097351e-02 2.88928830e-02]
[1.00000000e+00 3.32940087e-01 6.34082325e-01 1.65221728e-01
 1.10849102e-01 2.11111425e-01 5.50089365e-02 4.02060395e-01
 1.04764177e-01 2.72982194e-02]
[1.00000000e+00 2.40806764e-01 8.01868014e-01 5.65565133e-01
 5.79878976e-02 1.93095242e-01 1.36191910e-01 6.42992311e-01
 4.53508590e-01 3.19863920e-01]
[1.00000000e+00 1.47645882e-01 9.15990339e-01 4.83812330e-01
 2.17993064e-02 1.35242201e-01 7.14328982e-02 8.39038301e-01
 4.43167420e-01 2.34074371e-01]
[1.00000000e+00 9.65588256e-01 1.22158294e-01 5.50512785e-01
 9.32360680e-01 1.17954614e-01 5.31568680e-01 1.49226487e-02
 6.72497026e-02 3.03064327e-01]
[1.00000000e+00 9.50039684e-01 9.96815010e-01 9.11407288e-01
 9.02575402e-01 9.47013817e-01 8.65873092e-01 9.93640165e-01
 9.08504465e-01 8.30663244e-01]
[1.00000000e+00 8.46235321e-02 6.28532668e-01 2.93755563e-01
 7.16114219e-03 5.31886544e-02 2.48586333e-02 3.95053315e-01
 1.84634968e-01 8.62923309e-02]
[1.00000000e+00 3.56615326e-01 7.23538516e-01 8.34060068e-01
 1.27174490e-01 2.58024924e-01 2.97438603e-01 5.23507985e-01
 6.03474584e-01 6.95656197e-01]
[1.00000000e+00 5.98121586e-01 2.39109308e-01 5.63000589e-01
 3.57749431e-01 1.43016438e-01 3.36742805e-01 5.71732610e-02
 1.34618681e-01 3.16969664e-01]
[1.00000000e+00 3.17143237e-02 8.68126339e-01 4.33944619e-01
 1.00579833e-03 2.75320398e-02 1.37622601e-02 7.53643341e-01
 3.76718753e-01 1.88307932e-01]
[1.00000000e+00 6.04362339e-01 8.79087434e-01 9.20987455e-01
 3.65253837e-01 5.31287338e-01 5.56610132e-01 7.72794717e-01
 8.09628499e-01 8.48217892e-01]
[1.00000000e+00 7.81231089e-01 2.41581610e-01 7.68088671e-01
 6.10322014e-01 1.88731064e-01 6.00054749e-01 5.83616742e-02
 1.85556098e-01 5.89960207e-01]
[1.00000000e+00 9.43640011e-01 5.87979156e-01 6.29856063e-02
 8.90456470e-01 5.54840657e-01 5.94357382e-02 3.45719488e-01
 3.70342236e-02 3.96718659e-03]
[1.00000000e+00 1.85586980e-01 4.14177481e-01 6.88608737e-01
 3.44425271e-02 7.68659478e-02 1.27796816e-01 1.71542986e-01
 2.85206232e-01 4.74181992e-01]
[1.00000000e+00 3.64850349e-01 5.58881646e-01 8.12312767e-01
 1.33115777e-01 2.03908164e-01 2.96372596e-01 3.12348695e-01
 4.53986697e-01 6.59852031e-01]
[1.00000000e+00 7.44247951e-01 2.36635302e-01 4.69132127e-01
 5.53905013e-01 1.76115338e-01 3.49150625e-01 5.59962660e-02
 1.11013222e-01 2.20084953e-01]
[1.00000000e+00 9.80930054e-01 8.26536469e-01 9.34627016e-02
 9.62223772e-01 8.10774464e-01 9.16803729e-02 6.83162535e-01
 7.72503314e-02 8.73527659e-03]
[1.00000000e+00 9.31847847e-01 7.20718349e-02 4.20230570e-01
 8.68340409e-01 6.71599842e-02 3.91590952e-01 5.19434939e-03
 3.02867883e-02 1.76593732e-01]
[1.00000000e+00 8.97871392e-01 7.37892718e-02 6.08188027e-01
```

 8.06173036e-01 6.62532761e-02 5.46074630e-01 5.44485663e-03
 4.48777516e-02 3.69892677e-01]
[1.00000000e+00 1.85949658e-01 4.86407668e-01 5.87332618e-01
 3.45772753e-02 9.04473395e-02 1.09214300e-01 2.36592419e-01
 2.85683089e-01 3.44959605e-01]
[1.00000000e+00 4.80216051e-01 1.37201177e-01 2.10558953e-01
 2.30607456e-01 6.58862073e-02 1.01113789e-01 1.88241629e-02
 2.88889361e-02 4.43350727e-02]
[1.00000000e+00 1.93956598e-01 2.67151329e-01 5.79425025e-01
 3.76191620e-02 5.18157629e-02 1.12383307e-01 7.13698325e-02
 1.54794165e-01 3.35733360e-01]
[1.00000000e+00 8.62985699e-01 7.73062255e-01 4.55666112e-02
 7.44744317e-01 6.67141671e-01 3.93233339e-02 5.97625250e-01
 3.52258273e-02 2.07631606e-03]
[1.00000000e+00 8.20177338e-01 7.96694601e-02 8.69210276e-01
 6.72690866e-01 6.53430857e-02 7.12906570e-01 6.34722287e-03
 6.92495133e-02 7.55526503e-01]
[1.00000000e+00 8.22227931e-01 2.22206096e-01 6.61740361e-01
 6.76058770e-01 1.82704058e-01 5.44101407e-01 4.93755491e-02
 1.47042742e-01 4.37900305e-01]
[1.00000000e+00 6.96952676e-01 1.91209521e-01 2.12524679e-01
 4.85743033e-01 1.33263987e-01 1.48119644e-01 3.65610810e-02
 4.06367421e-02 4.51667392e-02]
[1.00000000e+00 7.49248663e-01 8.00397658e-01 5.42419523e-01
 5.61373560e-01 5.99696876e-01 4.06407103e-01 6.40636411e-01
 4.34151316e-01 2.94218939e-01]
[1.00000000e+00 6.65721061e-01 1.01275337e-01 1.43845237e-01
 4.43184531e-01 6.74211249e-02 9.57608038e-02 1.02566939e-02
 1.45679749e-02 2.06914522e-02]
[1.00000000e+00 6.80878857e-02 4.53324651e-01 5.89939495e-01
 4.63596018e-03 3.08659170e-02 4.01677329e-02 2.05503239e-01
 2.67434116e-01 3.48028608e-01]
[1.00000000e+00 5.89626569e-01 6.70886414e-02 2.46344398e-01
 3.47659491e-01 3.95572455e-02 1.45251202e-01 4.50088580e-03
 1.65269110e-02 6.06855624e-02]
[1.00000000e+00 8.89933893e-01 2.01767511e-02 4.63880320e-01
 7.91982333e-01 1.79559747e-02 4.12822819e-01 4.07101287e-04
 9.35959777e-03 2.15184951e-01]
[1.00000000e+00 7.39343636e-01 5.20477268e-01 6.47738604e-01
 5.46629012e-01 3.84811556e-01 4.78901415e-01 2.70896587e-01
 3.37133219e-01 4.19565299e-01]
[1.00000000e+00 9.61225862e-01 5.62270992e-01 5.67859814e-01
 9.23955158e-01 5.40469419e-01 5.45841539e-01 3.16148669e-01
 3.19291101e-01 3.22464768e-01]
[1.00000000e+00 4.02749352e-01 4.09549128e-01 3.70202045e-01
 1.62207041e-01 1.64945646e-01 1.49098634e-01 1.67730488e-01
 1.51615925e-01 1.37049554e-01]
[1.00000000e+00 7.19543192e-02 2.71132409e-01 5.79390543e-02
 5.17742405e-03 1.95091479e-02 4.16896520e-03 7.35127833e-02
 1.57091554e-02 3.35693401e-03]
[1.00000000e+00 6.27181523e-01 8.98446989e-01 1.61405156e-01
 3.93356662e-01 5.63489351e-01 1.01230331e-01 8.07206992e-01
 1.45013976e-01 2.60516242e-02]
[1.00000000e+00 4.00125861e-01 7.83332980e-01 2.46257704e-01
 1.60100704e-01 3.13431783e-01 9.85340758e-02 6.13610557e-01
 1.92901781e-01 6.06428568e-02]
[1.00000000e+00 1.44831851e-01 8.47462852e-01 3.14399776e-01
 2.09762651e-02 1.22739614e-01 4.55351016e-02 7.18193286e-01
 2.66442131e-01 9.88472192e-02]
[1.00000000e+00 2.16035958e-01 5.35499107e-01 3.63064516e-02
 4.66715350e-02 1.15687063e-01 7.84349904e-03 2.86759294e-01
 1.94420724e-02 1.31815843e-03]
[1.00000000e+00 9.10721842e-03 8.47590559e-01 9.13817199e-02
 8.29414274e-05 7.71919235e-03 8.32233283e-04 7.18409756e-01

```
 7.74542830e-02 8.35061873e-03]
[1.00000000e+00 2.42478670e-01 3.23404982e-01 4.51893915e-01
 5.87959054e-02 7.84188099e-02 1.09574635e-01 1.04590782e-01
 1.46144743e-01 2.04208110e-01]
[1.00000000e+00 9.58672279e-01 7.79185543e-01 6.76718746e-01
 9.19052539e-01 7.46983580e-01 6.48751503e-01 6.07130110e-01
 5.27289463e-01 4.57948261e-01]
[1.00000000e+00 2.12264003e-01 6.43505091e-02 4.39743641e-01
 4.50560070e-02 1.36592967e-02 9.33417455e-02 4.14098802e-03
 2.82977272e-02 1.93374470e-01]
[1.00000000e+00 4.94040183e-01 4.57294313e-02 6.70697008e-01
 2.44075702e-01 2.25921766e-02 3.31351273e-01 2.09118089e-03
 3.06705928e-02 4.49834477e-01]
[1.00000000e+00 3.27570294e-01 5.35530007e-01 1.78755714e-01
 1.07302298e-01 1.75423722e-01 5.85550618e-02 2.86792388e-01
 9.57290485e-02 3.19536052e-02]
[1.00000000e+00 7.42212061e-01 7.21228428e-01 9.10760352e-01
 5.50878744e-01 5.35304438e-01 6.75977318e-01 5.20170445e-01
 6.56866257e-01 8.29484419e-01]
[1.00000000e+00 9.18129202e-01 1.02577606e-01 4.43107778e-01
 8.42961231e-01 9.41794952e-02 4.06830191e-01 1.05221652e-02
 4.54529350e-02 1.96344503e-01]
[1.00000000e+00 5.03180681e-01 1.07019067e-01 9.14920232e-02
 2.53190798e-01 5.38499268e-02 4.60370186e-02 1.14530806e-02
 9.79139093e-03 8.37079032e-03]
[1.00000000e+00 8.48799407e-01 4.40314721e-01 9.63121193e-01
 7.20460433e-01 3.73738874e-01 8.17496697e-01 1.93877054e-01
 4.24076439e-01 9.27602432e-01]
[1.00000000e+00 8.88401653e-01 9.58641021e-01 1.72250171e-01
 7.89257497e-01 8.51658268e-01 1.53027336e-01 9.18992608e-01
 1.65126080e-01 2.96701213e-02]
[1.00000000e+00 4.10551994e-02 9.77771907e-01 2.76849199e-01
 1.68552940e-03 4.01426206e-02 1.13660991e-02 9.56037902e-01
 2.70695369e-01 7.66454791e-02]
[1.00000000e+00 7.37928417e-01 8.68033163e-01 7.15665220e-01
 5.44538348e-01 6.40546338e-01 5.28109703e-01 7.53481572e-01
 6.21221145e-01 5.12176708e-01]
[1.00000000e+00 4.54802519e-01 3.44222623e-01 2.22807464e-01
 2.06845331e-01 1.56553316e-01 1.01333396e-01 1.18489214e-01
 7.66953696e-02 4.96431660e-02]]
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Loading [MathJax]/extensions/Safe.js