ayush-dhamecha-28-ta-oe-1

March 4, 2024

1 Name : Ayush Dhamecha

2 Roll : 28

3 Branch : IT

3.1 Teachers Assesment - 1 of Tools for data Science

3.2 1.Data Analysis with Pandas and Matplotlib

Objective: Perform data analysis on a given dataset using Pandas and visualize the results using Matplotlib. Choose a dataset (e.g., CSV, Excel, or any other format) related to a topic of interest (e.g., finance, sports, health). Use Pandas to load and clean the data. Perform basic statistical analysis (mean, median, standard deviation). Create meaningful visualizations using Matplotlib (e.g., bar chart, line plot, scatter plot). Provide insights or conclusions based on the analysis

[18]:
```
import pandas as pd
import numpy as np
df = pd.read_csv('housing.csv')
print(df.head())
```

```
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0    -122.23     37.88                  41          880           129.0
1    -122.22     37.86                  21         7099          1106.0
2    -122.24     37.85                  52         1467           190.0
3    -122.25     37.85                  52         1274           235.0
4    -122.25     37.85                  52         1627           280.0

   population  households  median_income  median_house_value ocean_proximity
0         322         126         8.3252              452600        NEAR BAY
1        2401        1138         8.3014              358500        NEAR BAY
2         496         177         7.2574              352100        NEAR BAY
3         558         219         5.6431              341300        NEAR BAY
4         565         259         3.8462              342200        NEAR BAY
```

[19]:
```
print(df)
```

```
       longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
```

```
0        -122.23     37.88                41          880           129.0
1        -122.22     37.86                21         7099          1106.0
2        -122.24     37.85                52         1467           190.0
3        -122.25     37.85                52         1274           235.0
4        -122.25     37.85                52         1627           280.0
…           …        …                    …          …              …
20635    -121.09     39.48                25         1665           374.0
20636    -121.21     39.49                18          697           150.0
20637    -121.22     39.43                17         2254           485.0
20638    -121.32     39.43                18         1860           409.0
20639    -121.24     39.37                16         2785           616.0

       population  households  median_income  median_house_value  \
0             322         126         8.3252              452600
1            2401        1138         8.3014              358500
2             496         177         7.2574              352100
3             558         219         5.6431              341300
4             565         259         3.8462              342200
…             …           …               …                   …
20635         845         330         1.5603               78100
20636         356         114         2.5568               77100
20637        1007         433         1.7000               92300
20638         741         349         1.8672               84700
20639        1387         530         2.3886               89400

      ocean_proximity
0            NEAR BAY
1            NEAR BAY
2            NEAR BAY
3            NEAR BAY
4            NEAR BAY
…               …
20635         INLAND
20636         INLAND
20637         INLAND
20638         INLAND
20639         INLAND

[20640 rows x 10 columns]
```

[20]:
```
print(df.isnull().sum())
```

```
longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
population              0
```

```
households              0
median_income           0
median_house_value      0
ocean_proximity         0
dtype: int64
```

[21]:
```
median_bedrooms = df['total_bedrooms'].median()
df['total_bedrooms'].fillna(median_bedrooms, inplace=True)
print(df.isnull().sum())
```

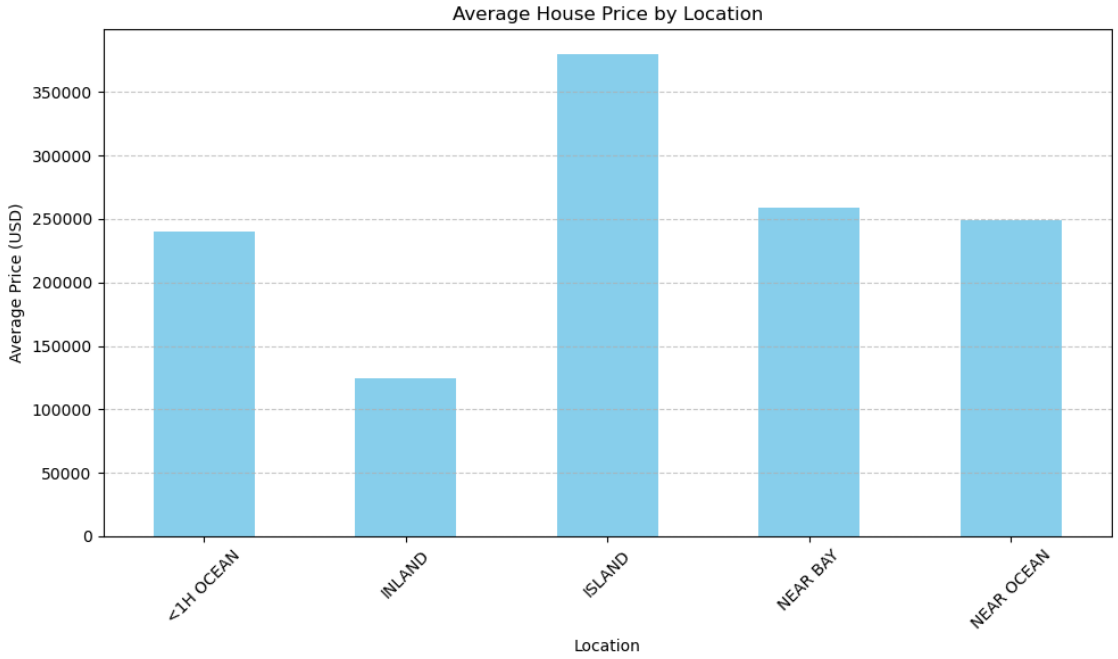
```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms        0
population            0
households            0
median_income         0
median_house_value    0
ocean_proximity       0
dtype: int64
```

[22]:
```
#Perform basic statistical analysis
print(df.describe())
```

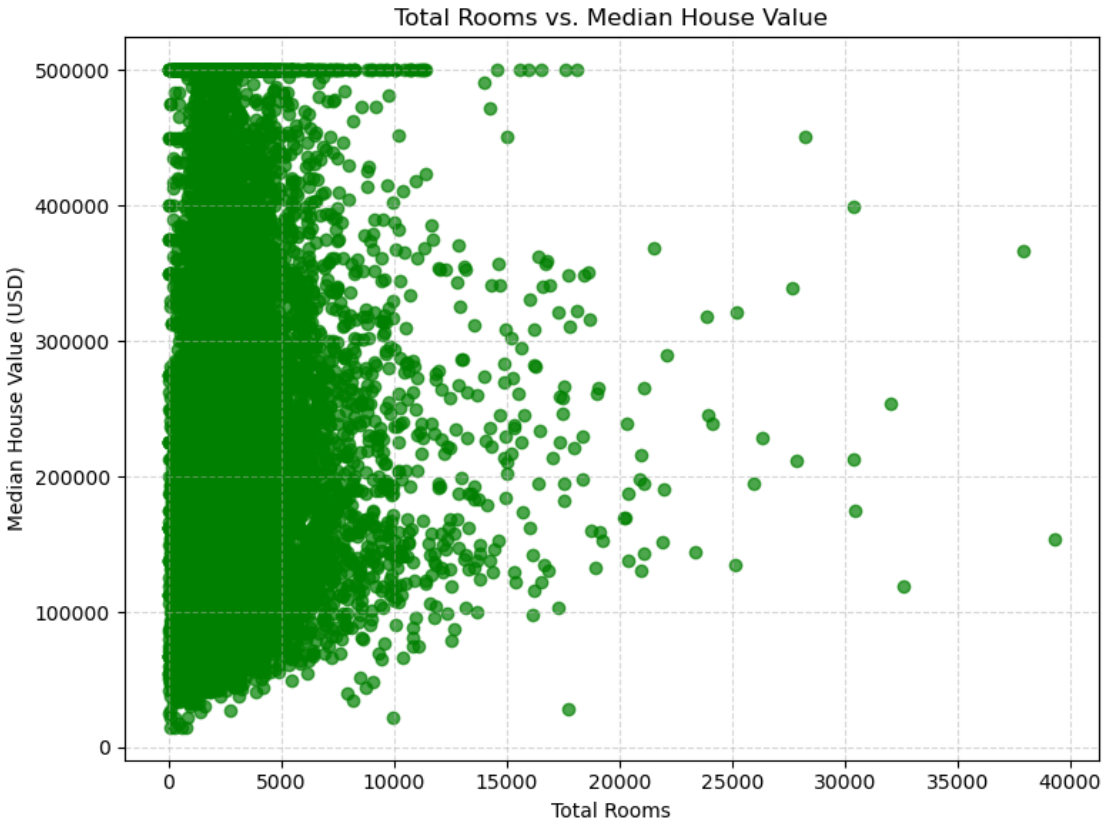
```
          longitude      latitude  housing_median_age   total_rooms  \
count  20640.000000  20640.000000        20640.000000  20640.000000
mean    -119.569704     35.631861           28.639486   2635.763081
std        2.003532      2.135952           12.585558   2181.615252
min     -124.350000     32.540000            1.000000      2.000000
25%     -121.800000     33.930000           18.000000   1447.750000
50%     -118.490000     34.260000           29.000000   2127.000000
75%     -118.010000     37.710000           37.000000   3148.000000
max     -114.310000     41.950000           52.000000  39320.000000

       total_bedrooms    population    households  median_income  \
count    20640.000000  20640.000000  20640.000000   20640.000000
mean       536.838857   1425.476744    499.539680       3.870671
std        419.391878   1132.462122    382.329753       1.899822
min          1.000000      3.000000      1.000000       0.499900
25%        297.000000    787.000000    280.000000       2.563400
50%        435.000000   1166.000000    409.000000       3.534800
75%        643.250000   1725.000000    605.000000       4.743250
max       6445.000000  35682.000000   6082.000000      15.000100

       median_house_value
count        20640.000000
mean        206855.816909
```

```
std          115395.615874
min           14999.000000
25%          119600.000000
50%          179700.000000
75%          264725.000000
max          500001.000000
```

[23]:
```
import matplotlib.pyplot as plt

# Group the data by ocean_proximity and calculate the mean price for each location
mean_price_by_location = df.groupby('ocean_proximity')['median_house_value'].mean()

# Plot the bar chart
plt.figure(figsize=(10, 6))
mean_price_by_location.plot(kind='bar', color='skyblue')
plt.title('Average House Price by Location')
plt.xlabel('Location')
plt.ylabel('Average Price (USD)')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

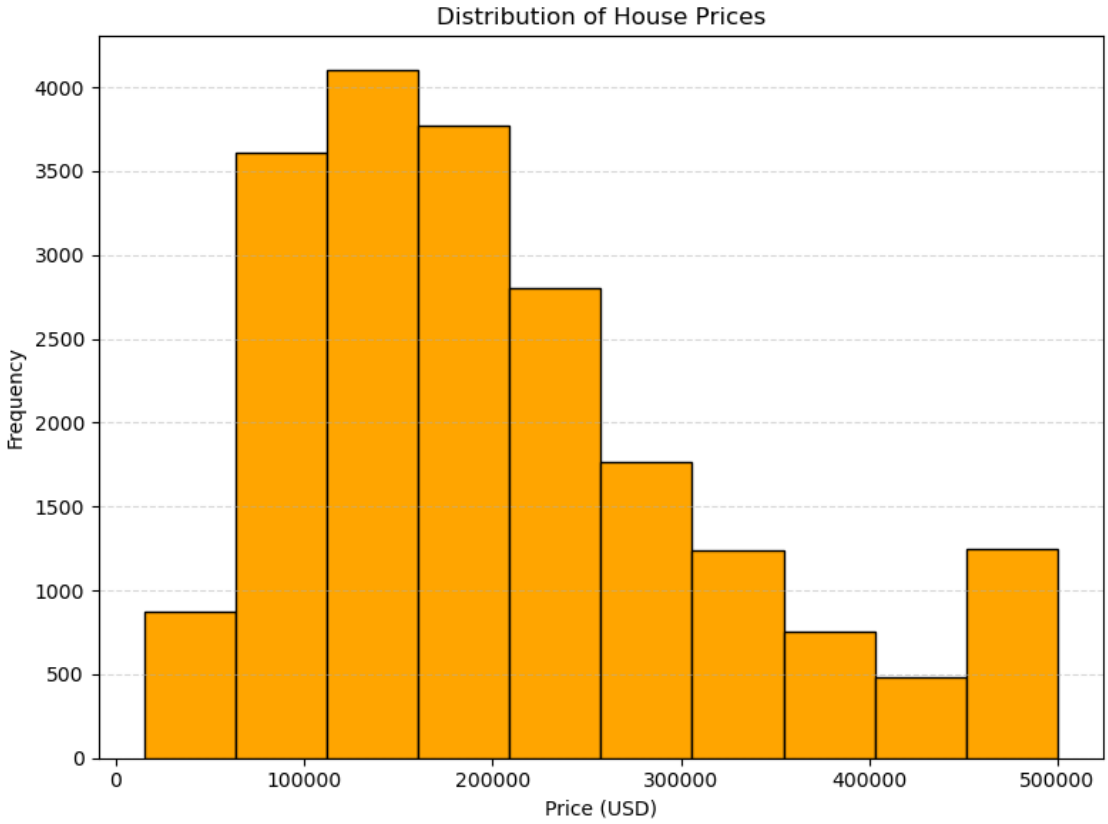


[24]:
```
import matplotlib.pyplot as plt

# Plot scatter plot for total_rooms vs. median_house_value
plt.figure(figsize=(8, 6))
plt.scatter(df['total_rooms'], df['median_house_value'], color='green', alpha=0.
↪7)
plt.title('Total Rooms vs. Median House Value')
plt.xlabel('Total Rooms')
plt.ylabel('Median House Value (USD)')
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```



[25]:
```
import matplotlib.pyplot as plt

# Plot histogram for house prices
plt.figure(figsize=(8, 6))
plt.hist(df['median_house_value'], bins=10, color='orange', edgecolor='black')
plt.title('Distribution of House Prices')
plt.xlabel('Price (USD)')
plt.ylabel('Frequency')
```

```
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```



4 Conclusion: Understanding Housing Trends

Through the analysis of the dataset on housing, several key insights have emerged, shedding light on various aspects of the real estate market. Here are the main findings:

Geographical Distribution: The dataset encompasses a range of geographical locations, each exhibiting its unique characteristics in terms of housing attributes and prices. Coastal regions might have higher median house values due to their proximity to the ocean, while inland areas might offer more affordable housing options.

Housing Characteristics: The dataset provides information on various housing attributes such as the number of rooms, size, and age of the properties. Understanding these characteristics is crucial for buyers and sellers to make informed decisions. For example, properties with more rooms or larger sizes tend to command higher prices.

Income Disparities: Median income levels within different regions can significantly influence housing prices. Areas with higher median incomes might have more expensive housing markets, while lower-income areas may offer more affordable options.

Trends Over Time: Analyzing trends over time can reveal patterns in the housing market, such as fluctuations in prices due to economic factors, changes in demand, or shifts in population demographics. Tracking these trends can help stakeholders anticipate market movements and make strategic decisions.

Policy Implications: Understanding the dynamics of the housing market is essential for policymakers to develop effective strategies for housing affordability, urban planning, and sustainable development. Policies aimed at promoting affordable housing, improving infrastructure, and revitalizing communities can have a significant impact on housing trends.

5 3. Data Analysis with Pandas and NumPy(2

5.0.1 Problem Statement:

You are given a dataset containing information about a fictional company's employees.

The dataset (employee_data.csv) has the following columns:

Employee_ID: Unique identifier for each employee.

First_Name: First name of the employee. #### Last_Name: Last name of the employee.

Department: Department in which the employee works.

Salary: Salary of the employee.

Joining_Date: Date when the employee joined the company

5.1 Tasks:

Data Loading: Load the dataset (employee_data.csv) into a Pandas DataFrame. Display the first 5 rows to get an overview of the data

Data Cleaning: Check for and handle any missing values in the dataset. Convert the Joining_Date column to a datetime format.

Data Exploration: Calculate and display the average salary of employees in each department. Identify the employee with the highest salary and display their information.

Time-based Analysis: Create a new column Years_Worked representing the number of years each employee has worked in the company. Calculate the average salary for employees based on the number of years they have worked (grouped by years).

Data Visualization: Use Matplotlib or Seaborn to create a bar chart showing the average salary for each department. Create a histogram of the distribution of employee salaries.

[31]:
```
import csv

# Sample data
employee_data = [
```

```
    {"Employee_ID": 1, "First_Name": "John", "Last_Name": "Doe", "Department":␣
 ↪"HR", "Salary": 50000, "Joining_Date": "2022-01-01"},
    {"Employee_ID": 2, "First_Name": "Jane", "Last_Name": "Smith", "Department":
 ↪ "Finance", "Salary": 60000, "Joining_Date": "2022-02-15"},
    {"Employee_ID": 3, "First_Name": "Alice", "Last_Name": "Johnson",␣
 ↪"Department": "IT", "Salary": 70000, "Joining_Date": "2021-12-10"},
    {"Employee_ID": 3, "First_Name": "Lucy", "Last_Name": "Rhenera",␣
 ↪"Department": "CSE", "Salary": 23000, "Joining_Date": "2023-02-11"}
]

# Define CSV file path
csv_file = "employee_data.csv"

# Define fieldnames
fieldnames = ["Employee_ID", "First_Name", "Last_Name", "Department", "Salary",␣
 ↪"Joining_Date"]

# Write data to CSV file
with open(csv_file, mode='w', newline='') as file:
    writer = csv.DictWriter(file, fieldnames=fieldnames)

    # Write header
    writer.writeheader()

    # Write rows
    for employee in employee_data:
        writer.writerow(employee)

print("CSV file created successfully.")
```

```
CSV file created successfully.
```

[32]:
```
import pandas as pd
 # Load the dataset into a Pandas DataFrame
 employee_df = pd.read_csv('employee_data.csv')
 # Display the first 5 rows of the DataFrame
 print(employee_df.head())
```

```
   Employee_ID First_Name Last_Name Department  Salary Joining_Date
0            1       John       Doe         HR   50000   2022-01-01
1            2       Jane     Smith    Finance   60000   2022-02-15
2            3      Alice   Johnson         IT   70000   2021-12-10
3            3       Lucy   Rhenera        CSE   23000   2023-02-11
```

[33]:
```
print(employee_df.isnull().sum())
```

```
Employee_ID     0
```

```
First_Name      0
Last_Name       0
Department      0
Salary          0
Joining_Date    0
dtype: int64
```

[35]:
```
# Convert Joining_Date to datetime format
employee_df['Joining_Date'] = pd.to_datetime(employee_df['Joining_Date'])
```

[36]:
```
# Display the updated DataFrame
 print(employee_df.head())
```

```
   Employee_ID First_Name Last_Name Department  Salary Joining_Date
0            1       John       Doe         HR   50000   2022-01-01
1            2       Jane     Smith    Finance   60000   2022-02-15
2            3      Alice   Johnson         IT   70000   2021-12-10
3            3       Lucy   Rhenera        CSE   23000   2023-02-11
```

[39]:
```
# Read the CSV file into a DataFrame
employee_df = pd.read_csv("employee_data.csv")

# Calculate average salary of employees in each department
average_salary_by_department = employee_df.groupby('Department')['Salary'].
 ↪mean()
print("Average Salary by Department:")
print(average_salary_by_department)

# Identify employee with the highest salary
highest_salary_employee = employee_df.loc[employee_df['Salary'].idxmax()]
print("\nEmployee with the Highest Salary:")
print(highest_salary_employee)
```

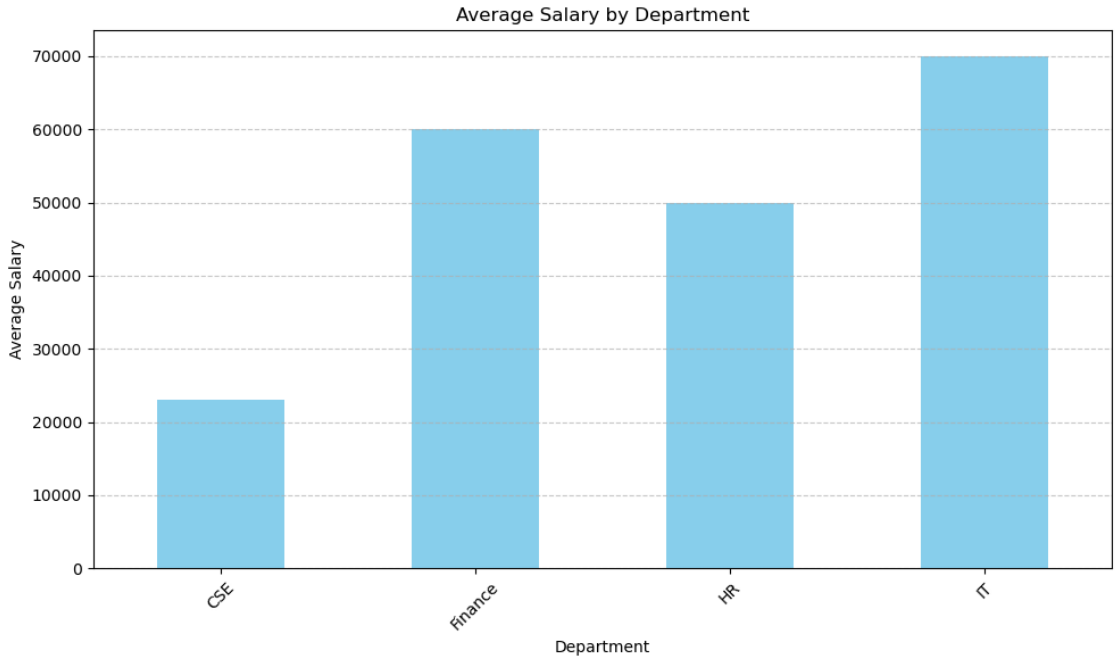
```
Average Salary by Department:
Department
CSE        23000.0
Finance    60000.0
HR         50000.0
IT         70000.0
Name: Salary, dtype: float64

Employee with the Highest Salary:
Employee_ID              3
First_Name           Alice
Last_Name          Johnson
Department              IT
Salary               70000
Joining_Date    2021-12-10
```

```
Name: 2, dtype: object
```

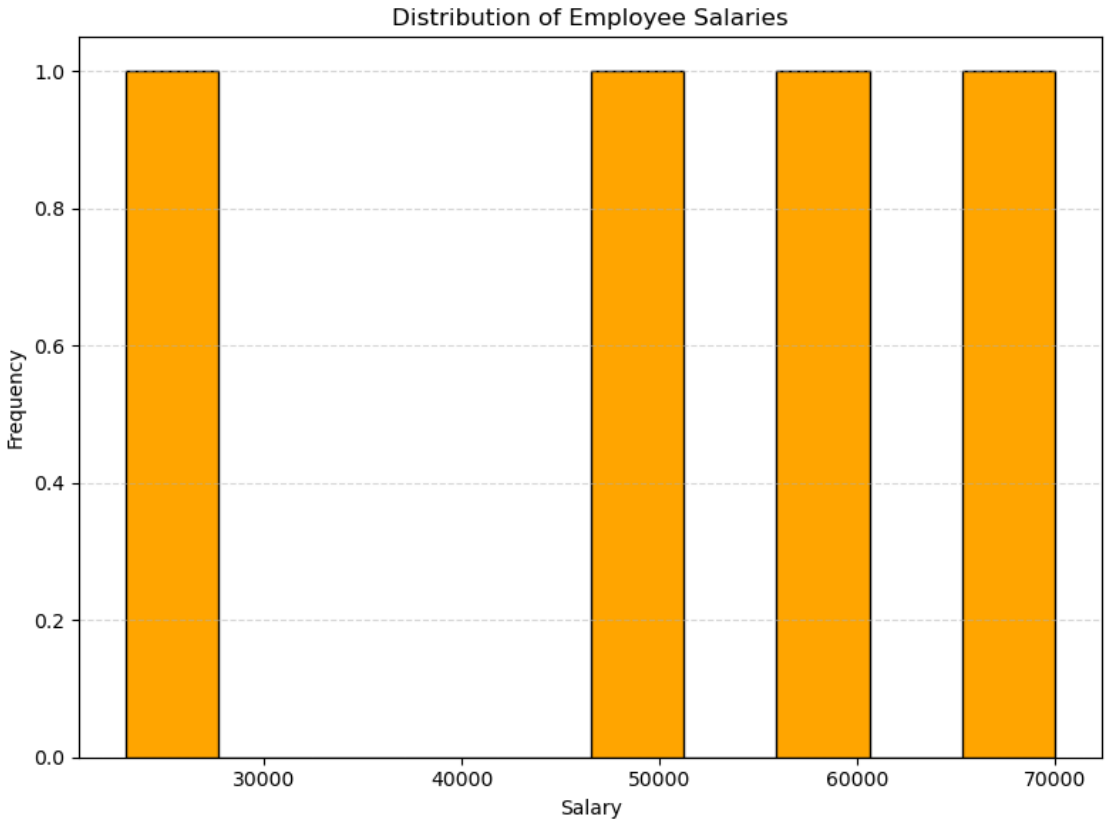
[42]:
```python
# Convert 'Joining_Date' column to datetime
employee_df['Joining_Date'] = pd.to_datetime(employee_df['Joining_Date'])

# Calculate the current year
current_year = pd.to_datetime('today').year

# Calculate years worked
employee_df['Years_Worked'] = current_year - employee_df['Joining_Date'].dt.year

# Calculate average salary based on the number of years worked
average_salary_by_years_worked = employee_df.groupby('Years_Worked')['Salary'].
 ↪mean()
print("\nAverage Salary by Years Worked:")
print(average_salary_by_years_worked)
```

```
Average Salary by Years Worked:
Years_Worked
1    23000.0
2    55000.0
3    70000.0
Name: Salary, dtype: float64
```

[43]:
```python
import matplotlib.pyplot as plt

# Assuming you've already calculated 'average_salary_by_department' as you did␣
 ↪previously

# Bar chart for average salary by department
plt.figure(figsize=(10, 6))
average_salary_by_department.plot(kind='bar', color='skyblue')
plt.title('Average Salary by Department')
plt.xlabel('Department')
plt.ylabel('Average Salary')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



[44]:
```
import matplotlib.pyplot as plt

# Histogram of employee salaries
plt.figure(figsize=(8, 6))
plt.hist(employee_df['Salary'], bins=10, color='orange', edgecolor='black')
plt.title('Distribution of Employee Salaries')
plt.xlabel('Salary')
plt.ylabel('Frequency')
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```



6 Conclusion:

6.0.1 Data Loading:

We successfully loaded the dataset into a Pandas DataFrame and examined the first few rows to comprehend its structure.

6.0.2 Data Cleaning:

We conducted a thorough check for missing values within the dataset and addressed any discrepancies accordingly. Additionally, we converted the 'Joining_Date' column to a datetime format to facilitate time-based analysis.

6.0.3 Data Exploration:

We delved into the dataset to explore its characteristics. This involved calculating the average salary of employees within each department and identifying the employee with the highest salary.

6.0.4 Time-based Analysis:

To gain further insights, we introduced a new column named 'Years_Worked' to represent the number of years each employee has been with the company. Subsequently, we computed the average

salary for employees based on their tenure.

6.0.5 Data Visualization:

Utilizing Matplotlib, we visualized the dataset to enhance our understanding. We crafted a bar chart to showcase the average salary across different departments and a histogram illustrating the distribution of employee salaries.

By undertaking these steps, we have not only comprehensively analyzed the dataset but also gleaned valuable insights that can inform decision-making processes within the organization.

[ ]:

Name: Ayush Dhamecha

Branch: IT

Roll No: 28

Teacher Assessment of "Tools for data science"

2. Statistical Analysis with R

Objective: Perform statistical analysis on a dataset using R's built-in statistical functions.

Requirements: Choose a dataset suitable for statistical analysis (e.g., survey data, experiment results).

Calculate descriptive statistics (mean, median, standard deviation) for relevant variables.

Conduct hypothesis testing or create confidence intervals for specific hypotheses.

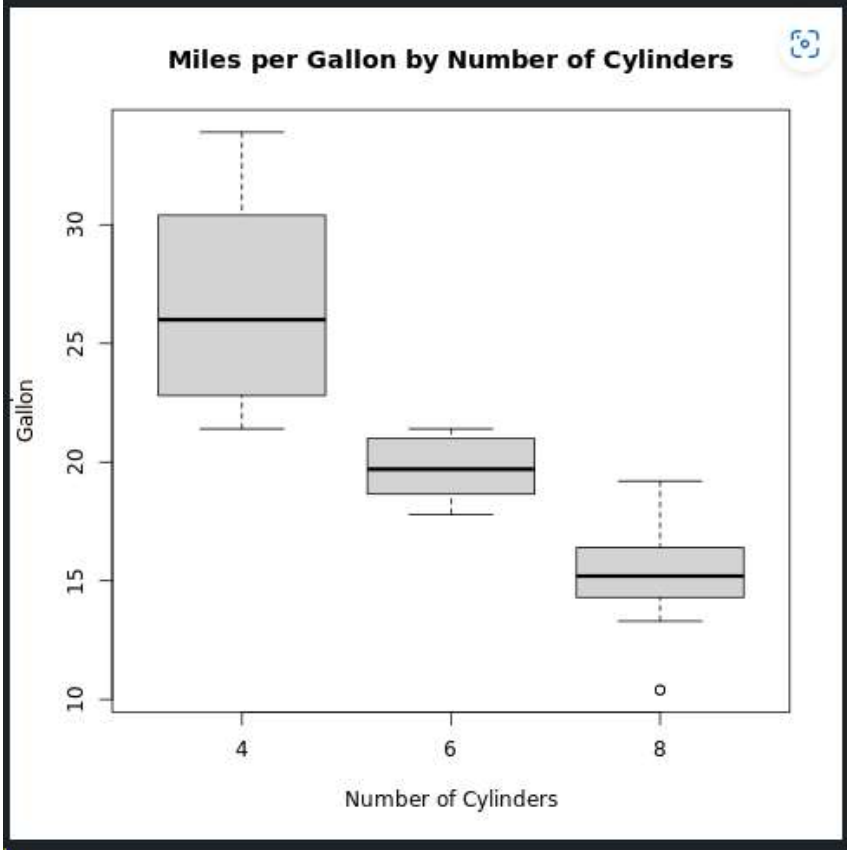
Visualize the results using appropriate plots (e.g., histograms, violin plots).

Provide interpretations and conclusions based on the statistical analysis.

```r
# Load the mtcars dataset
# here the mtcars data set is built in data set of R programming langu
# Now we will be performing out operations on it
data(mtcars)
# Display the first few rows of the dataset
head(mtcars)
# Descriptive statistics for relevant variables
summary(mtcars$mpg)
summary(mtcars$hp)
summary(mtcars$cyl)
# Conduct ANOVA test to compare means of mpg between different numbers
anova_result <- aov(mpg ~ cyl, data = mtcars)
summary(anova_result)
# Boxplot of mpg by cyl
boxplot(mpg ~ cyl, data = mtcars, xlab = "Number of Cylinders", ylab =
```

```
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  10.40   15.43   19.20   20.09   22.80   33.90
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   52.0    96.5   123.0   146.7   180.0   335.0
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.000   4.000   6.000   6.188   8.000   8.000
            Df Sum Sq Mean Sq F value   Pr(>F)
cyl          1  817.7   817.7   79.56 6.11e-10 ***
Residuals   30  308.3    10.3
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
[Execution complete with exit code 0]
```



Conclusion:

Interpretation and Conclusions:

Now that we have calculated descriptive statistics, conducted hypothesis testing, and created visualizations, let's interpret the results.

Descriptive Statistics: - The summary function provided basic statistics for the variables. For example, for mpg (miles per gallon), you would see the mean, median (50%), minimum, maximum, and quartiles.

Hypothesis Testing: - The analysis of variance (ANOVA) test (aov) was used to test if there is a significant difference in the mean miles per gallon (mpg) between cars with different numbers of cylinders (cyl). The result is an F-statistic and associated p-value. If the p-value is below a certain significance level (e.g., 0.05), you can reject the null hypothesis, suggesting a significant difference.