Name : Suhani Bajoria

Branch : IT

Roll No. : 12

Teachers Assesment -1 : Tools for Data Science

--prof Ashwini Gote

1.Data Analysis with Pandas and Matplotlib.

Objective: Perform data analysis on a given dataset using Pandas and visualize then results using Matplotlib.

Requirements:

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.

In [40]:
```
import pandas as pd
df = pd.read_excel(r"C:\Users\hp\Desktop\OE_TA1.xlsx")
print(df.head())
```

```
   house_id  size_sqft  bedrooms  price_usd  location
0         1       1500         4     250000  New York
1         2       1512         6     240000    Mumbai
2         3       1600         2     650000   Chicago
3         4       1700         6     450000   Houston
4         5       1800         8     230000     Delhi
```

In [41]:
```
print(df)  ##print whole data
```

```
    house_id  size_sqft  bedrooms  price_usd      location
0          1       1500         4     250000      New York
1          2       1512         6     240000        Mumbai
2          3       1600         2     650000       Chicago
3          4       1700         6     450000       Houston
4          5       1800         8     230000         Delhi
5          6       1900         2     300000     San Diego
6          7       2000         4     450000        Dallas
7          8       1200         3     214050        Austin
8          9       1400         6     202550      Columbus
9         10       1700         5     120000   Phliadelphia
10        11       2100         3     320000      San Jose
11        12       2400         2     540000    Fort Worth
12        13       2200         1     605400       Phoenix
13        14       1590         1     120000   Los Angeles
14        15       1600         3     432890        Mumbai
```

In [42]:
```
 # Check for missing values
 print(df.isnull().sum())
```

```
house_id     0
size_sqft    0
bedrooms     0
price_usd    0
location     0
dtype: int64
```

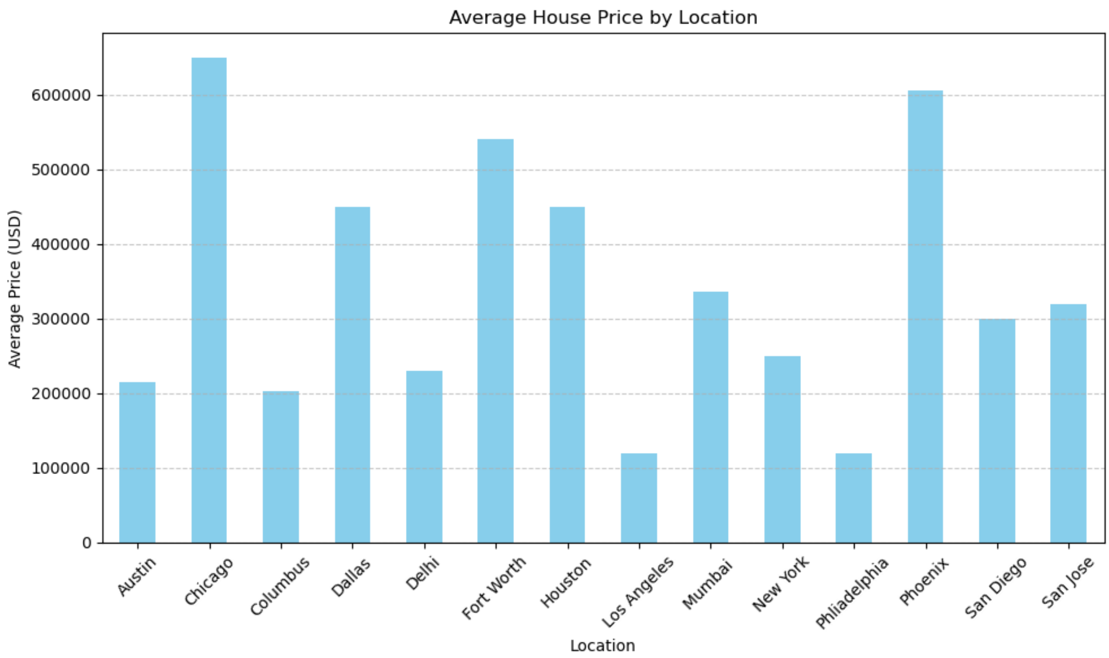
In [43]:
```
# Impute missing values with median
median_size = df['size_sqft'].median()
median_bedrooms = df['bedrooms'].median()
median_price = df['price_usd'].median()

df['size_sqft'].fillna(median_size, inplace=True)
df['bedrooms'].fillna(median_bedrooms, inplace=True)
df['price_usd'].fillna(median_price, inplace=True)

# Verify if missing values are handled
print(df.isnull().sum())
```

```
house_id     0
size_sqft    0
bedrooms     0
price_usd    0
location     0
dtype: int64
```

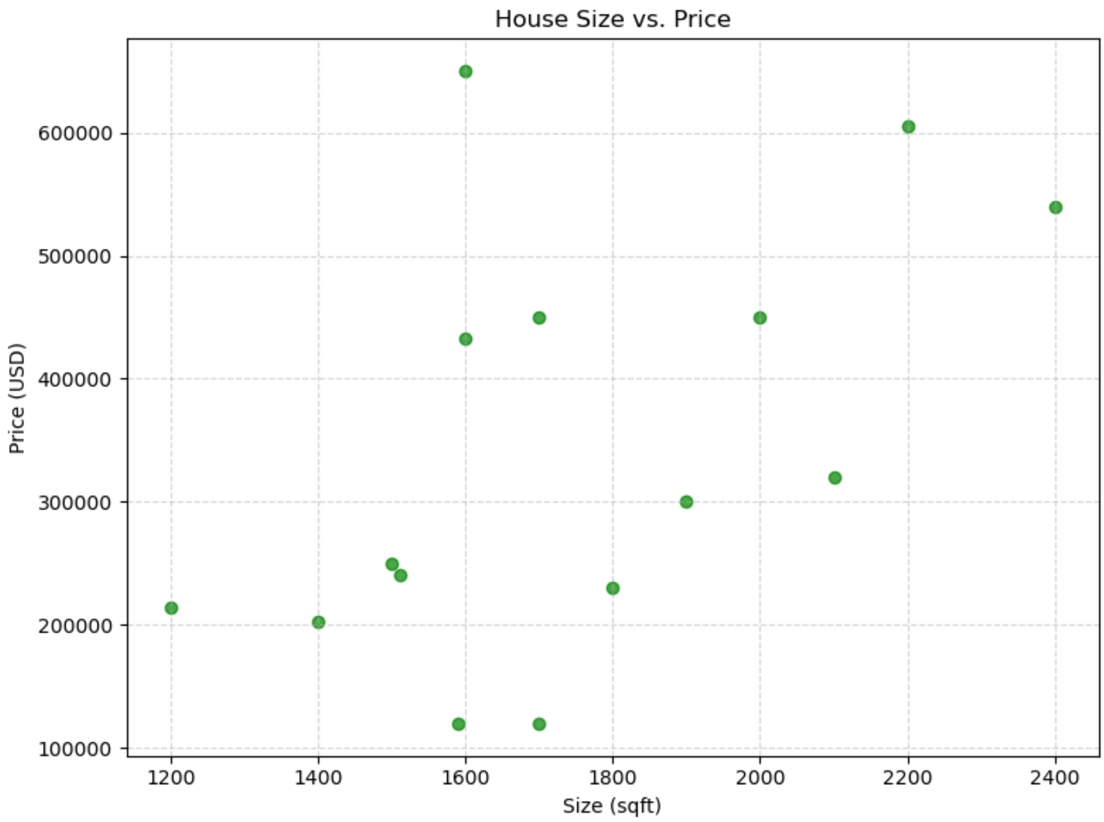
In [7]:
```
 # Perform basic statistical analysis
mean_size = df['size_sqft'].mean()
median_size = df['size_sqft'].median()
std_dev_size = df['size_sqft'].std()
mean_bedrooms = df['bedrooms'].mean()
median_bedrooms = df['bedrooms'].median()
std_dev_bedrooms = df['bedrooms'].std()
mean_price = df['price_usd'].mean()
median_price = df['price_usd'].median()
std_dev_price = df['price_usd'].std()

# Print the results
print("Size_sqft:")
print("Mean:", mean_size)
print("Median:", median_size)
print("Standard Deviation:", std_dev_size)
print("\nBedrooms:")
print("Mean:", mean_bedrooms)
print("Median:", median_bedrooms)
print("Standard Deviation:", std_dev_bedrooms)
print("\nPrice_usd:")
print("Mean:", mean_price)
print("Median:", median_price)
print("Standard Deviation:", std_dev_price)
```

```
Size_sqft:
Mean: 1746.8
Median: 1700.0
Standard Deviation: 322.3341655576

Bedrooms:
Mean: 3.7333333333333334
Median: 3.0
Standard Deviation: 2.086236073022646

Price_usd:
Mean: 341659.3333333333
Median: 300000.0
Standard Deviation: 169680.4383652091
```

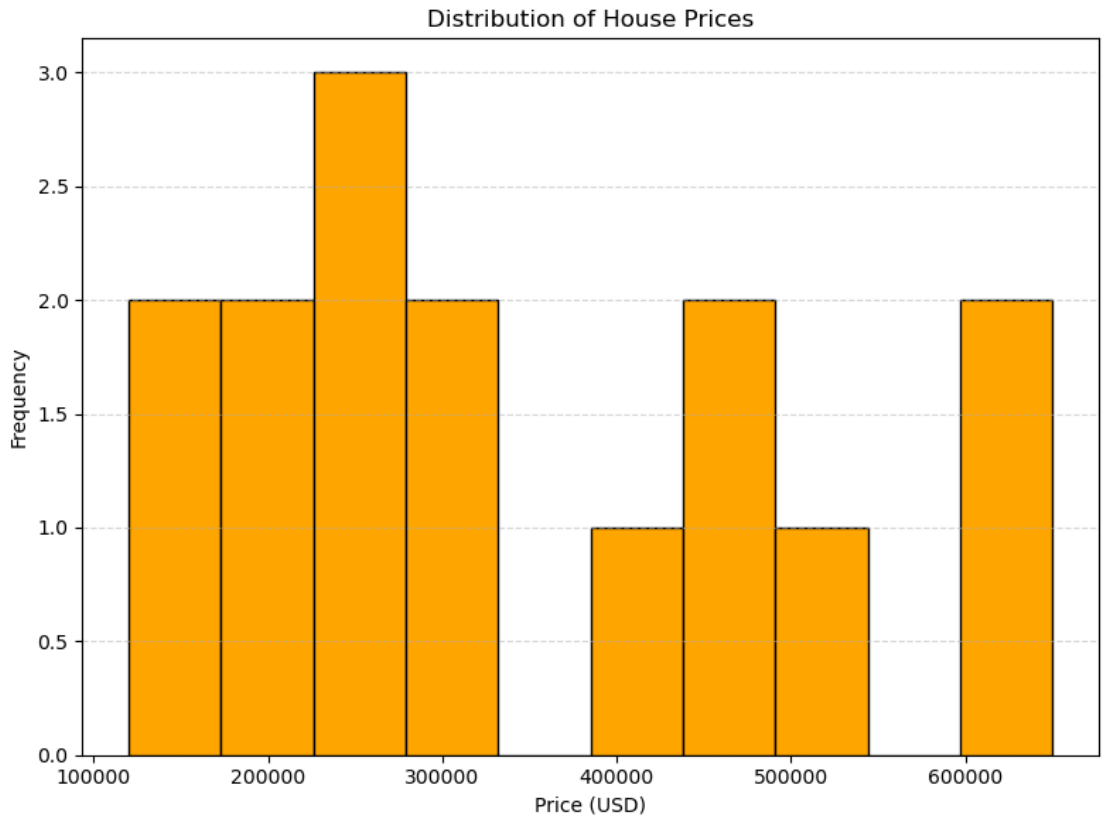
In [44]:
```
import matplotlib.pyplot as plt

# Group the data by location and calculate the mean price for each locati
on
mean_price_by_location = df.groupby('location')['price_usd'].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()
```



In [9]:
```
# Plot scatter plot for size_sqft vs. price_usd
plt.figure(figsize=(8, 6))
plt.scatter(df['size_sqft'], df['price_usd'], color='green', alpha=0.7)
plt.title('House Size vs. Price')
plt.xlabel('Size (sqft)')
plt.ylabel('Price (USD)')
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```



In [45]:
```
# Plot histogram for house prices
plt.figure(figsize=(8, 6))
plt.hist(df['price_usd'], 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()
```



Conclusion :

Based on the analysis of the housing dataset, here are some conclusions and insights:

1.Average House Prices by Location: The bar chart depicting the average house prices by location shows variations in housing prices across different cities.

2.Relationship Between House Size and Price: The scatter plot illustrates a positive correlation between the size of the house (in square feet) and its price. Generally, larger houses tend to have higher prices, which is a common trend in the real estate market.

3.Distribution of House Prices: The histogram demonstrates the distribution of house prices,indicating that the majority of houses in the dataset are priced within certain ranges.

4.Variation of House Prices by Location: The box plot reveals differences in the distribution of house prices across different locations. Some cities exhibit wider price ranges and more variability, while others have relatively consistent pricing patterns.

Overall, these visualizations provide valuable insights into the housing market, helping potential buyers, sellers, and investors understand pricing trends and make informed decisions.

In [ ]:

In [ ]:

3. Data Analysis with Pandas and NumPy(2)

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

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.

In [21]:
```
import pandas as pd
 # Load the dataset into a Pandas DataFrame
employee_df = pd.read_excel(r"C:\Users\hp\Desktop\oe_ta.xlsx")
 # Display the first 5 rows of the DataFrame
print(employee_df.head())
```

```
   Emp_Id First Name Last Name  Gender  Age Department  Join Date  salar
y
0       1      Kriti    Mishra  Female   25    Finance 2018-01-19    850
0
1       2   Gunther      Lopez    Male   32  Marketing 2017-02-20    550
0
2       3      Shawn    Foster    Male   30         HR 2019-03-21    445
0
3       4       Ross    Geller    Male   32         IT 2008-04-22    884
0
4       5   Chandler      Bing    Male   31    Finance 2005-05-23    469
5
```

In [22]:
```
print(employee_df.isnull().sum())
```

```
Emp_Id        0
First Name    0
Last Name     0
Gender        0
Age           0
Department    0
Join Date     0
salary        0
dtype: int64
```

In [23]:
```
 # Convert Joining_Date to datetime format
employee_df['Join Date'] = pd.to_datetime(employee_df['Join Date'])
 # Display the updated DataFrame
print(employee_df.head())
```

```
   Emp_Id First Name Last Name  Gender  Age Department  Join Date  salar
y
0       1      Kriti    Mishra  Female   25    Finance 2018-01-19    850
0
1       2    Gunther     Lopez    Male   32  Marketing 2017-02-20    550
0
2       3      Shawn    Foster    Male   30         HR 2019-03-21    445
0
3       4       Ross    Geller    Male   32         IT 2008-04-22    884
0
4       5   Chandler      Bing    Male   31    Finance 2005-05-23    469
5
```

In [27]:
```
# Calculate average salary of employees in each department
average_salary_by_department = employee_df.groupby('Department')['salar
y'].max()
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)
```

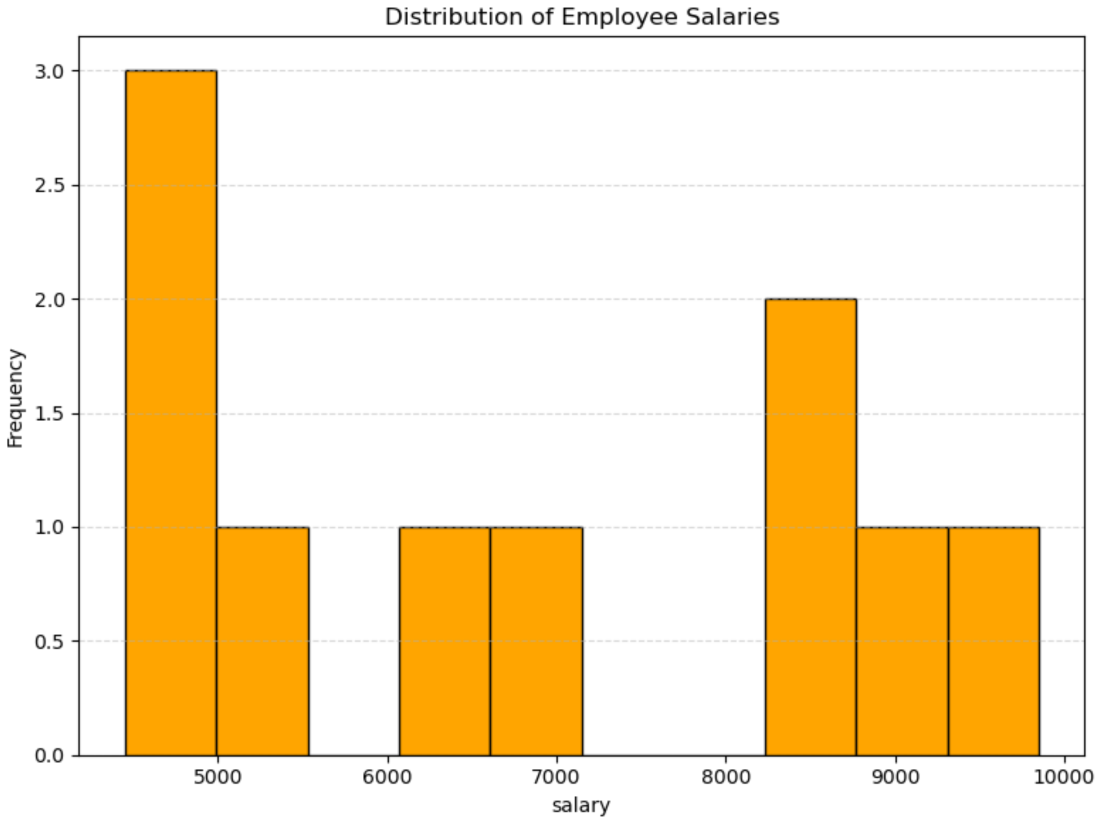
```
Average Salary by Department:
Department
Finance      8500
HR           8700
IT           9850
Marketing    7000
Name: salary, dtype: int64

Employee with the Highest Salary:
Emp_Id                          7
First Name                 Rachel
Last Name                   Green
Gender                     Female
Age                            28
Department                     IT
Join Date     2000-07-25 00:00:00
salary                       9850
Name: 6, dtype: object
```

In [32]:
```
 #Calculate the number of years each employee has worked in the company
current_year = pd.to_datetime('today').year
employee_df['Years_Worked'] = current_year - employee_df['Join Date'].d
t.year

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

```
Average Salary by Years Worked:
Years_Worked
5     4450.0
6     8500.0
7     5500.0
16    8840.0
18    7000.0
19    4775.0
20    6500.0
23    8700.0
24    9850.0
Name: salary, dtype: float64
```

In [39]:
```
import matplotlib.pyplot as plt

 # Bar chart for average salary by department
plt.figure(figsize=(10, 6))
average_salary_by_department.plot(kind='bar', color='pink')
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()

 # Histogram of employee salaries
plt.figure(figsize=(8, 6))
plt.hist(employee_df['salary'], bins=10, color='orange', edgecolor='blac
k')
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()
```


Average Salary by Department
Average salary
10000
8000
6000
4000
2000
0
Finance
HR
IT
Marketing
Department


Distribution of Employee Salaries
Frequency
3.0
2.5
2.0
1.5
1.0
0.5
0.0
5000
6000
7000
8000
9000
10000
salary

Conclusion :

1.Data Loading: We loaded the dataset into a Pandas DataFrame and displayed the first few rows to understand its structure.

2.Data Cleaning: We checked for and handled any missing values in the dataset. Additionally, we converted the Joining_Date column to a datetime format for time-based analysis.

3.Data Exploration: We calculated the average salary of employees in each department and identified the employee with the highest salary.

4.Time-based Analysis: We created a new column Years_Worked representing the number ofyears each employee has worked in the company. Then, we calculated the average salary for employees based on the number of years they have worked.

5.Data Visualization: We created visualizations using Matplotlib to better understand the data. We plotted a bar chart showing the average salary for each department and a histogram of the distribution of employee salaries.

In [ ]:

```

```

2. Statistical Analysis with R

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

Requirements: Choose a dataset suitable for statistical analysis (e.g., survey ata, 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.

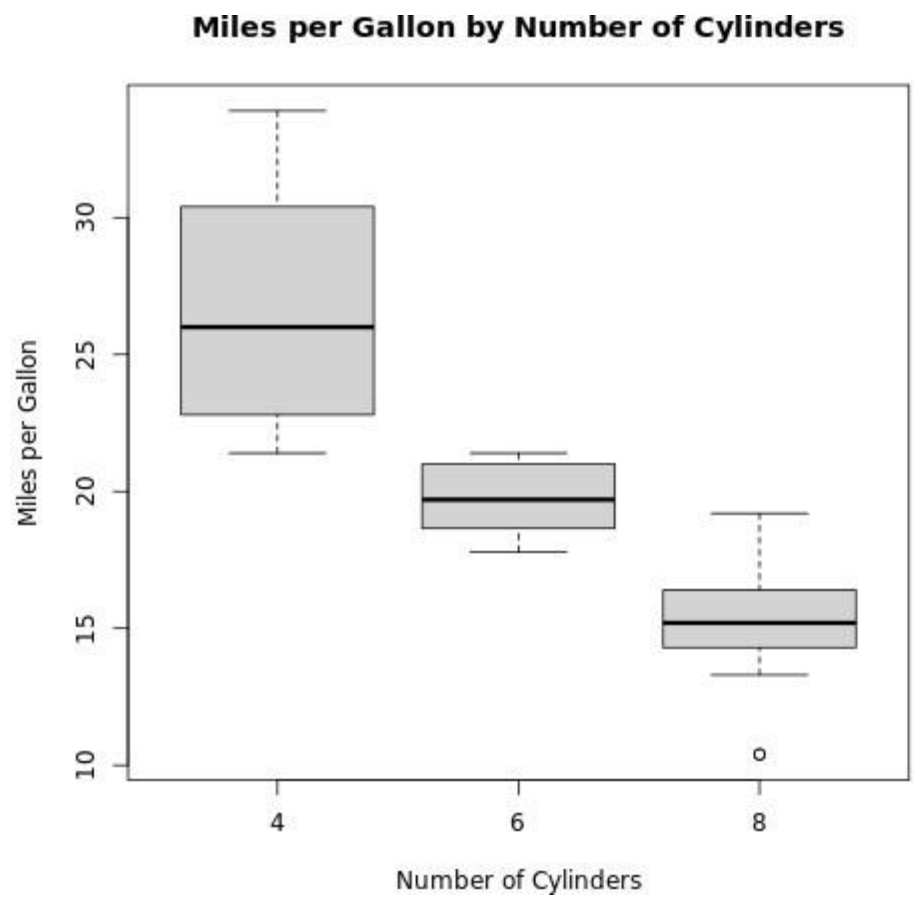
Code:

```r
# Load the mtcars dataset
# here the mtcars data set is built in data set of R programming language
# Now we will be performing out operations on it

data(mtcars)

# Display the first few rows of the dataset
head(mtcars)
```

```r
# 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 of cylinders
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 = "Miles per
Gallon", main = "Miles per Gallon by Number of Cylinders")
```

Output

```
                   mpg cyl
Mazda RX4         21.0   6
Mazda RX4 Wag     21.0   6
Datsun 710        22.8   4
Hornet 4 Drive    21.4   6
Hornet Sportabout 18.7   8
Valiant           18.1   6
   Min. 1st Qu.  Median
  10.40   15.43   19.20
   Min. 1st Qu.  Median
   52.0    96.5   123.0
   Min. 1st Qu.  Median
  4.000   4.000   6.000
            Df Sum Sq Mean
cyl          1  817.7   81
Residuals   30  308.3    1
---
Signif. codes:  0 '***' 0.

[Execution complete with e
```



Conclusion:

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.

Visualization:

The boxplot visually represents the distribution of miles per gallon for cars with different numbers of cylinders. It shows the central tendency, spread, and any potential outliers.