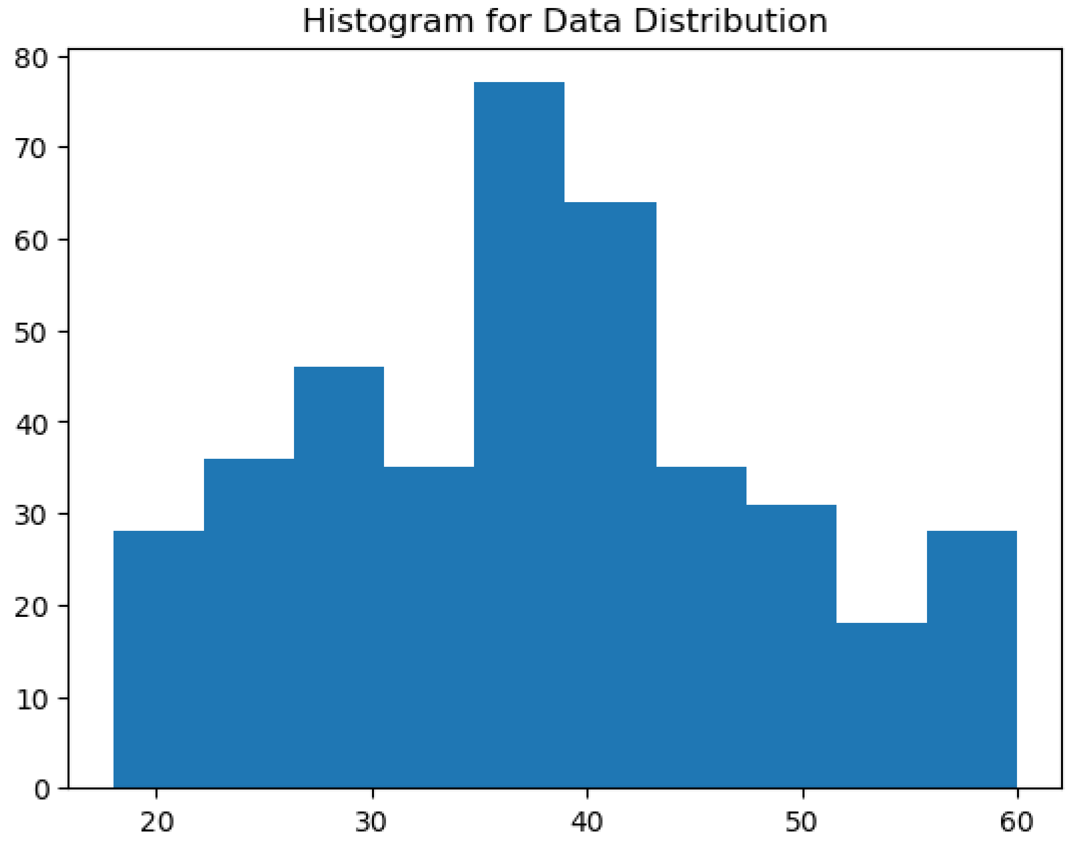
In [1]:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
# Generate dummy data
df = np.random.rand(100)
data = pd.read_csv("/home/c4leb/Desktop/C4LEB/Desktop/python_class/yafDataAna
# print(data)

# 1. Purpose of Data Visualization
# Data visualization simplifies data for easy understanding.
plt.hist(data['Age'], bins=10)
plt.title("Histogram for Data Distribution")
plt.show()

```



In [2]:

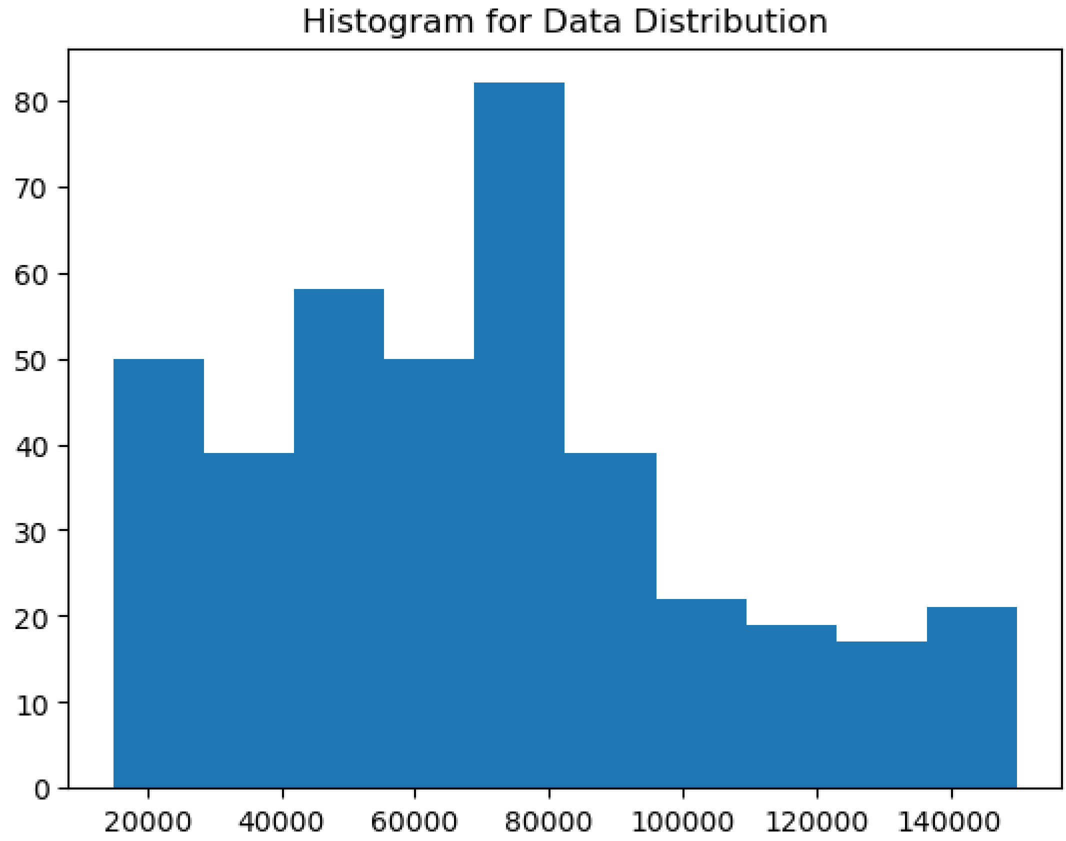
```
data ][""]
```

Out[2]:

```
      Age  EstimatedSalary  Purchased
  0   19.0         19000.0        0.0
  1   35.0         20000.0        0.0
  2   26.0         43000.0        0.0
  3   27.0         57000.0        0.0
  4   19.0         76000.0        0.0
...    ...             ...        ...
395   46.0         41000.0        1.0
396   51.0         23000.0        1.0
397   50.0         20000.0        1.0
398   36.0         33000.0        0.0
399   49.0         36000.0        1.0

400 rows × 3 columns
```

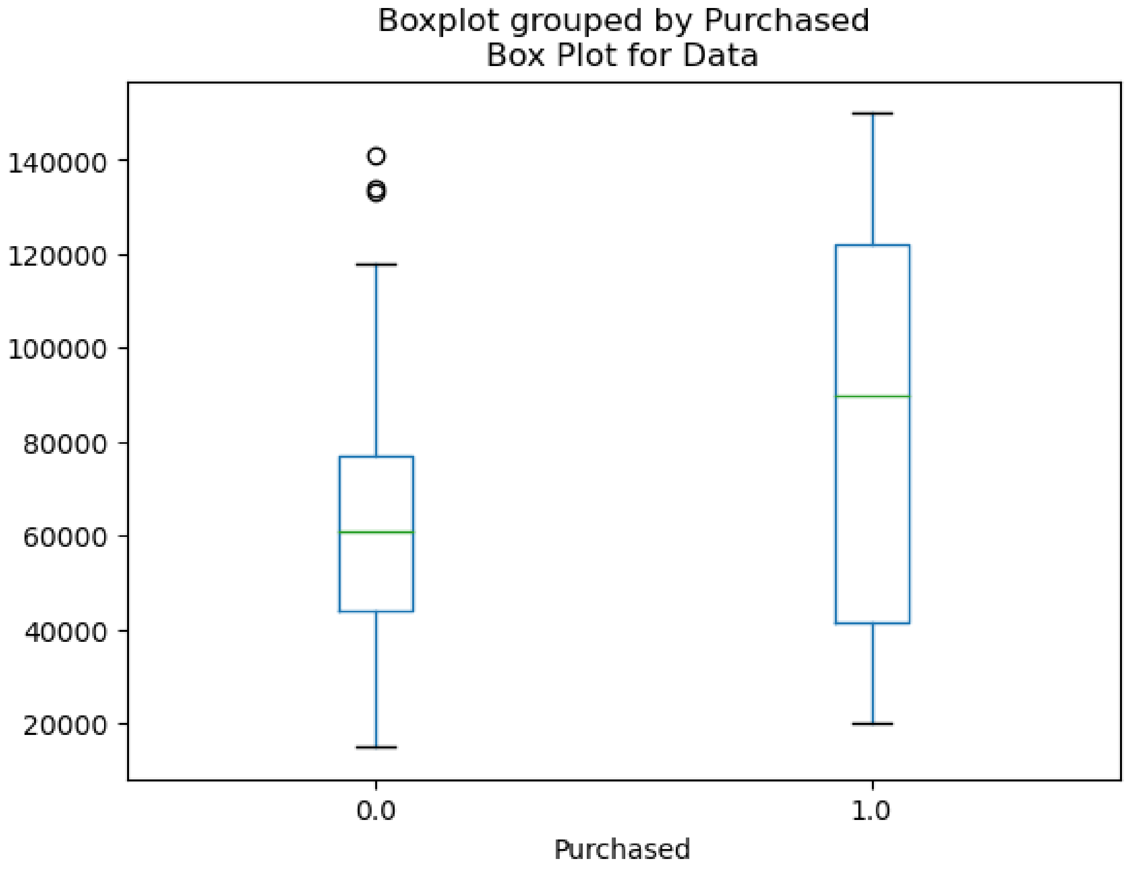
In [29]:

```python

# 2. Distribution of a Single Continuous Variable
# A histogram is suitable for showing the distribution of a single continuous
plt.hist(data["EstimatedSalary"], bins=10)
plt.title("Histogram for Data Distribution")
plt.show()

```


Histogram for Data Distribution

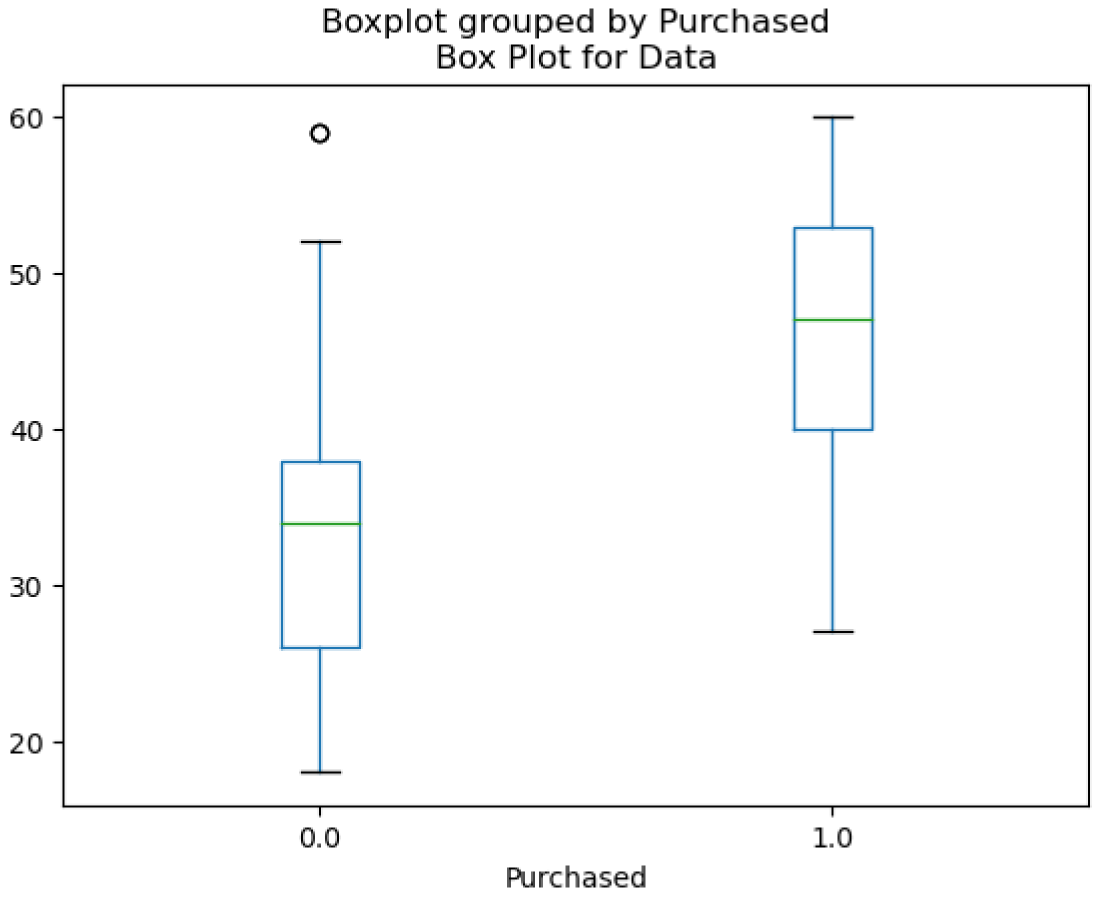
In [41]:

```

# 3. Box Plot Representation
# A box plot represents the interquartile range (IQR) of the data.
data.boxplot(by ='Purchased', column =['EstimatedSalary'], grid = False)
plt.title("Box Plot for Data")
plt.show()

```



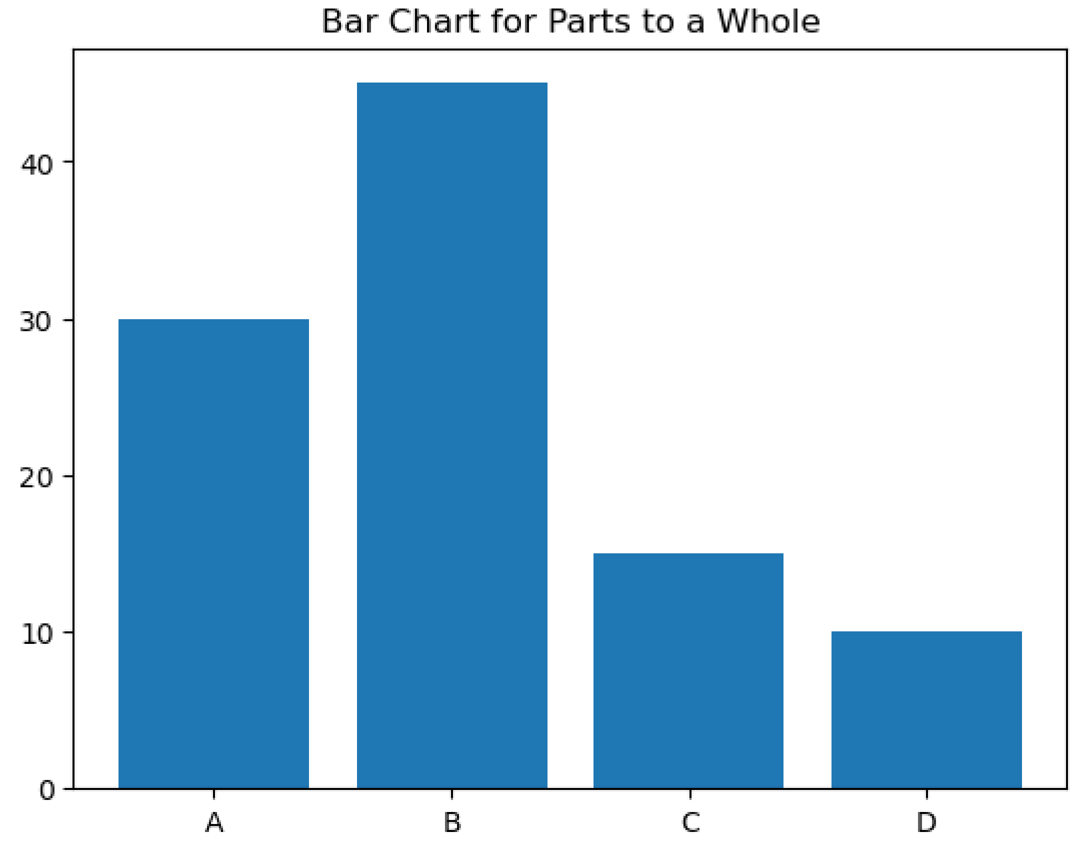
In [42]:

```

# 3. Box Plot Representation
# A box plot represents the interquartile range (IQR) of the data.
data.boxplot(by ='Purchased', column =['Age'], grid = False)
plt.title("Box Plot for Data")
plt.show()

```



In [4]:

```python
# 4. Comparing Parts to a Whole
# A bar chart is suitable for comparing parts to a whole.
categories = ['A', 'B', 'C', 'D']
values = [30, 45, 15, 10]
plt.bar(categories, values)
plt.title("Bar Chart for Parts to a Whole")
plt.show()

```

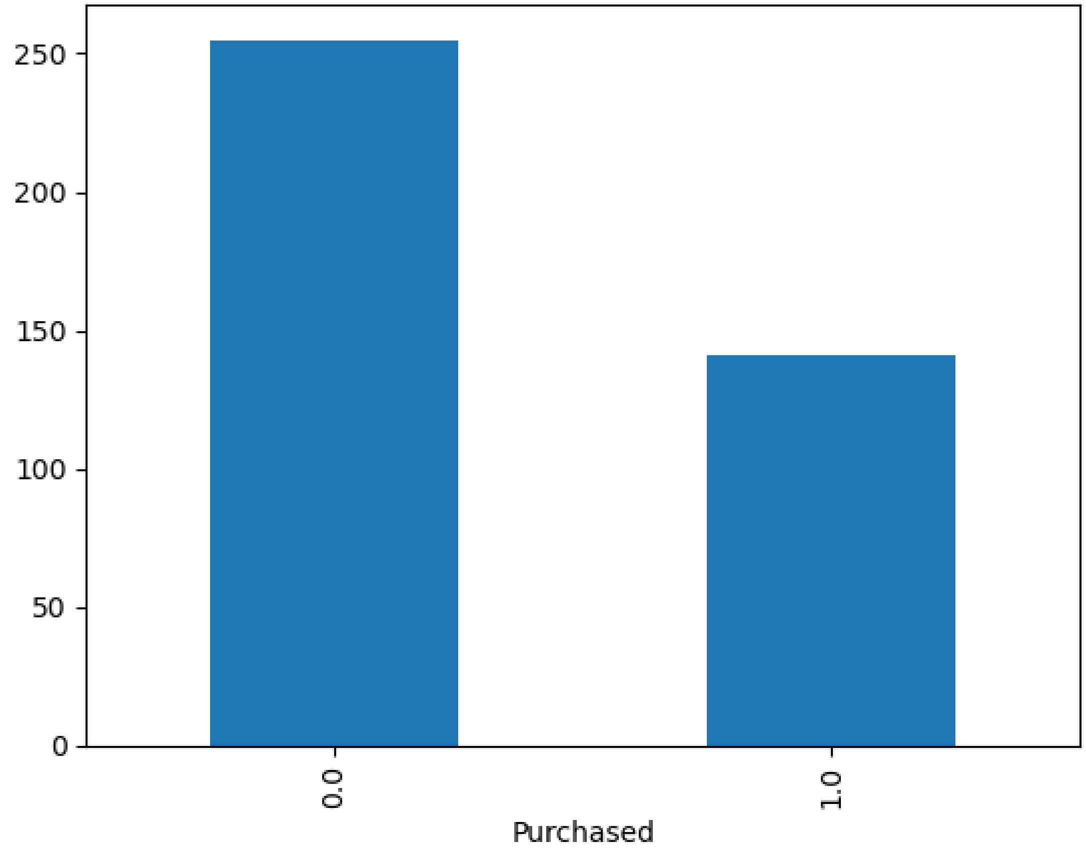


In [46]:

```
data['Purchased'].value_counts().plot(kind = "bar")
```

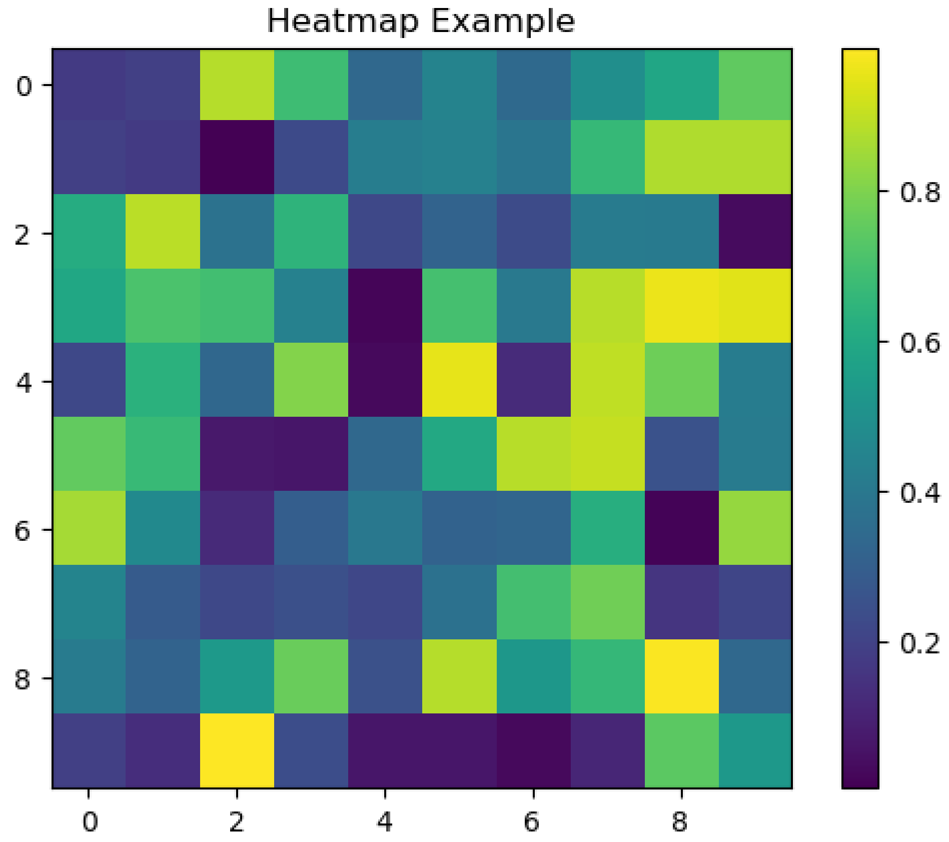
Out[46]:

```
<AxesSubplot:xlabel='Purchased'>
```



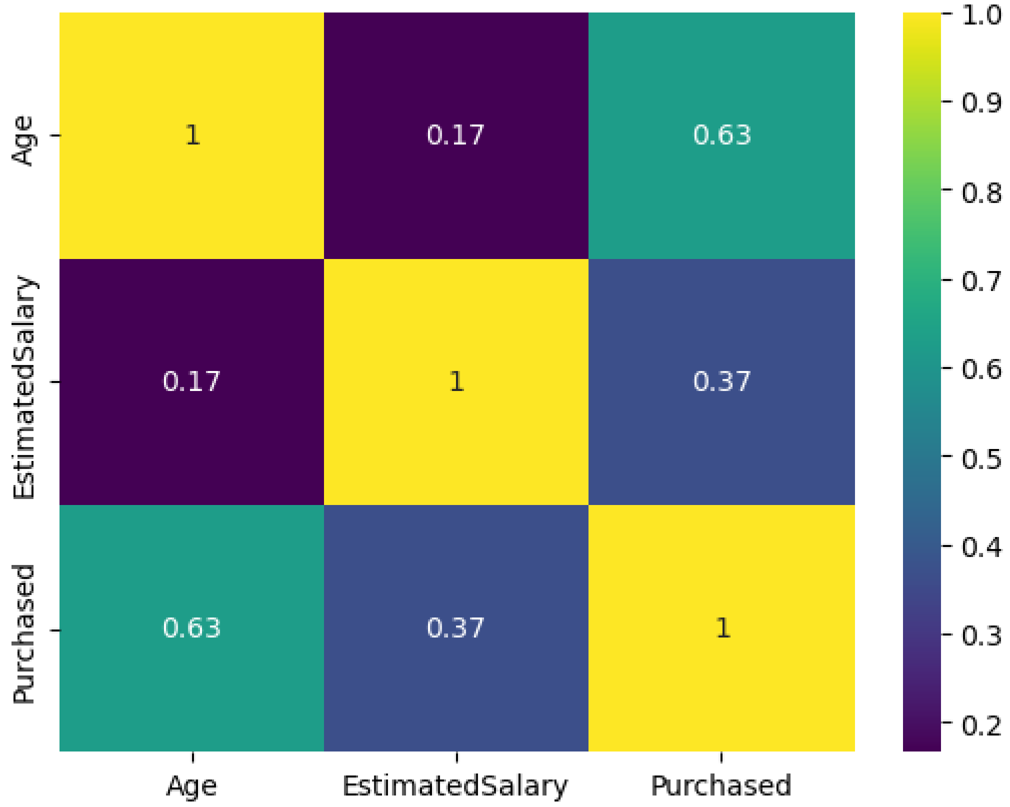
In [5]:

```
# 5. Heatmap
# A heatmap is a graphical representation using colors.
heatmap_data = np.random.rand(10, 10)
plt.imshow(heatmap_data, cmap='viridis')
plt.title("Heatmap Example")
plt.colorbar()
plt.show()



```



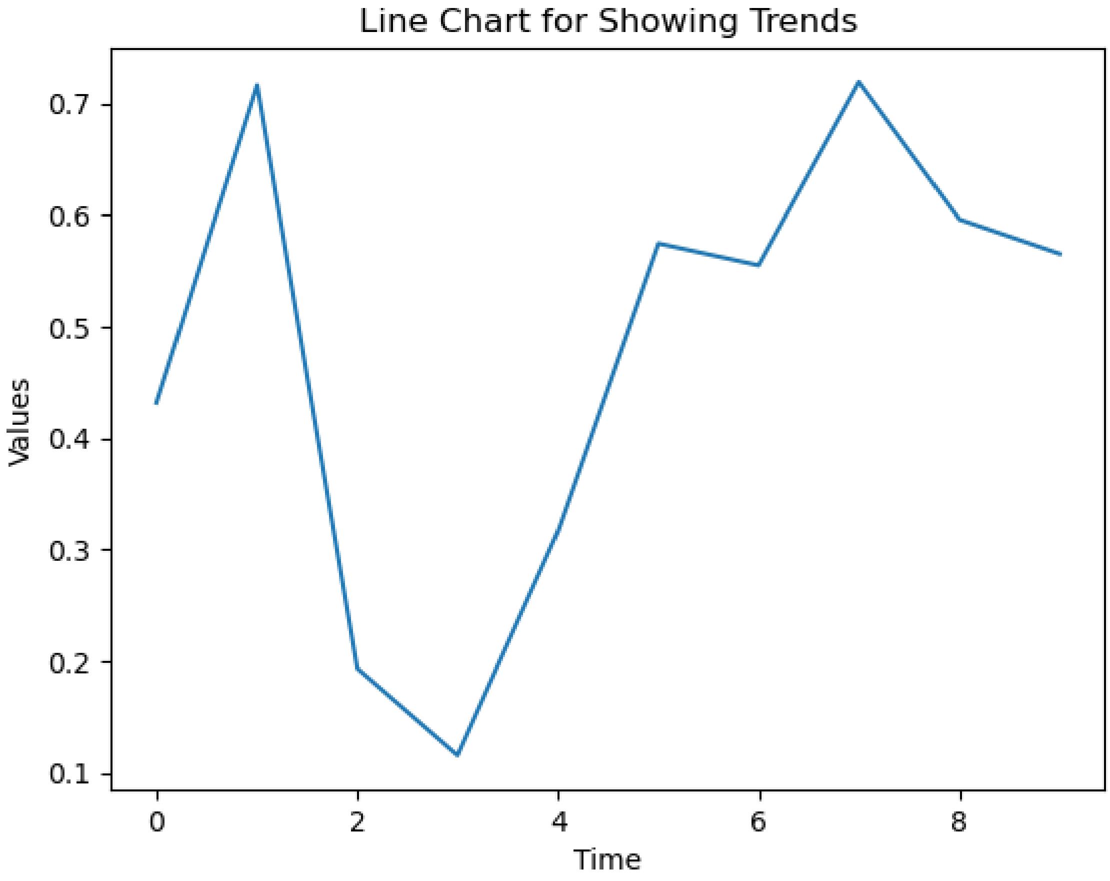
In [53]:

```python
import seaborn as sns
sns.heatmap(data.corr(), annot = True, cmap='viridis');
```



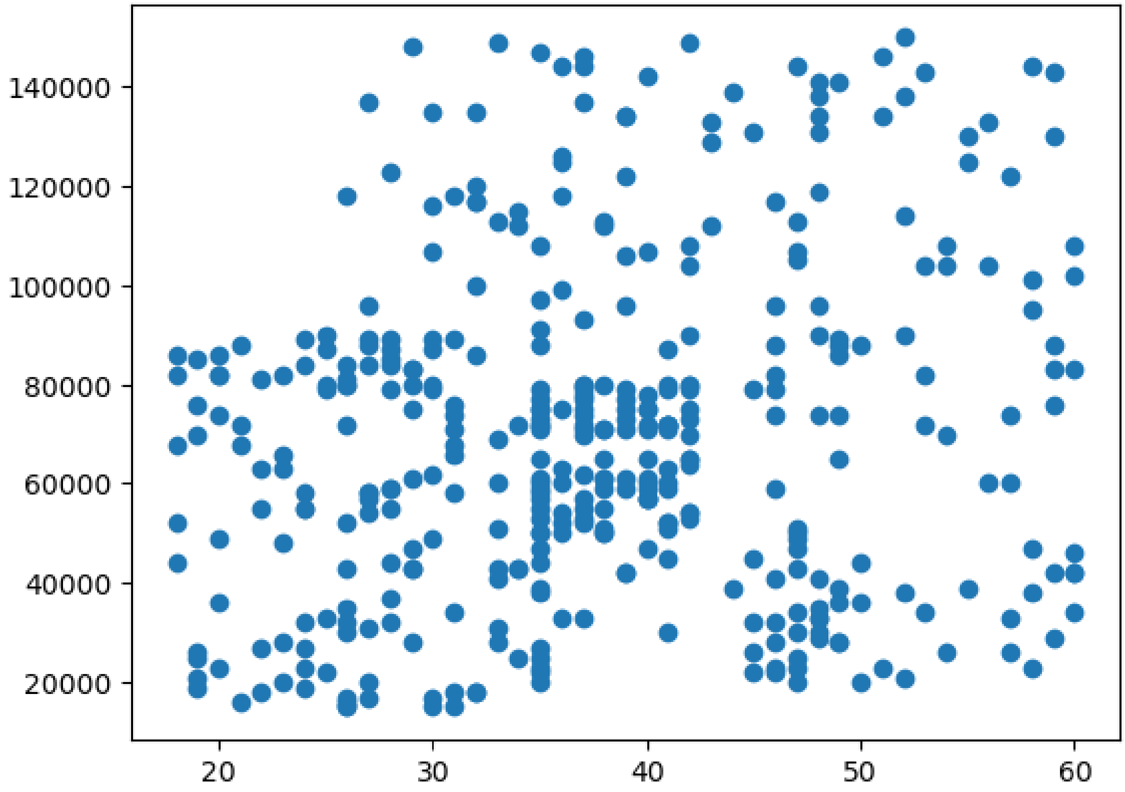
In [6]:

```python
# 6. Line Chart for Showing Trends
# Line charts are used for showing trends over time.
time = np.arange(0, 10, 1)
values = np.random.rand(10)
plt.plot(time, values)
plt.title("Line Chart for Showing Trends")
plt.xlabel("Time")
plt.ylabel("Values")
plt.show()

```



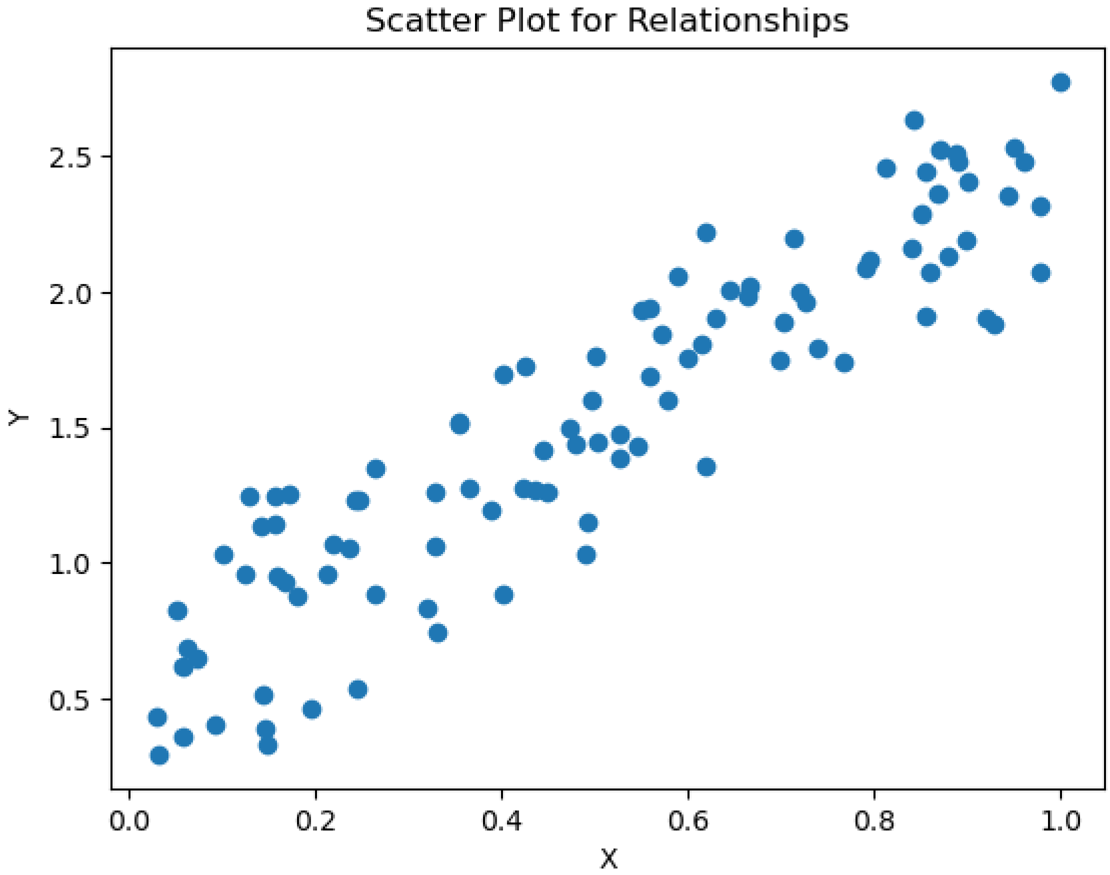
In [62]:

```
plt.scatter(data['Age'], data["EstimatedSalary"]);
```



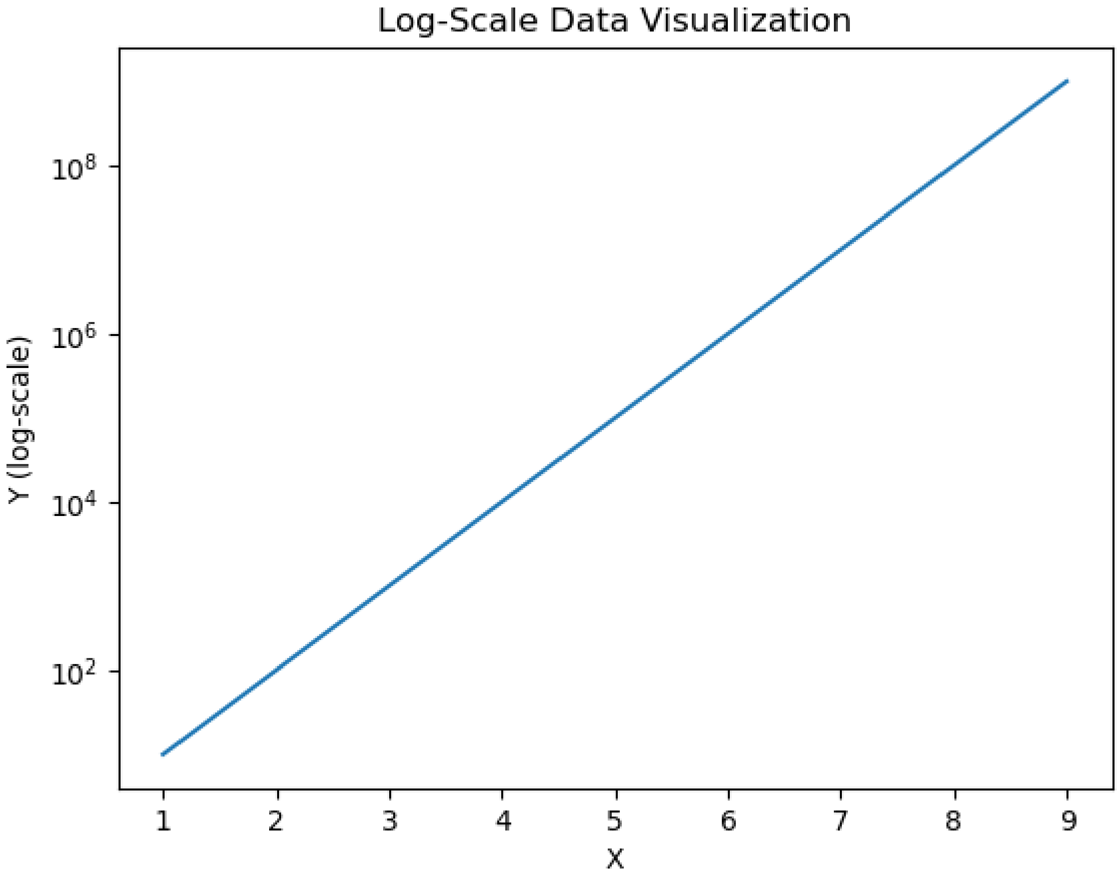
In [7]:

```python

# 7. Scatter Plot for Relationships
# A scatter plot shows relationships between two variables.
x = np.random.rand(100)
y = 2 * x + np.random.rand(100)
plt.scatter(x, y)
plt.title("Scatter Plot for Relationships")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()

```



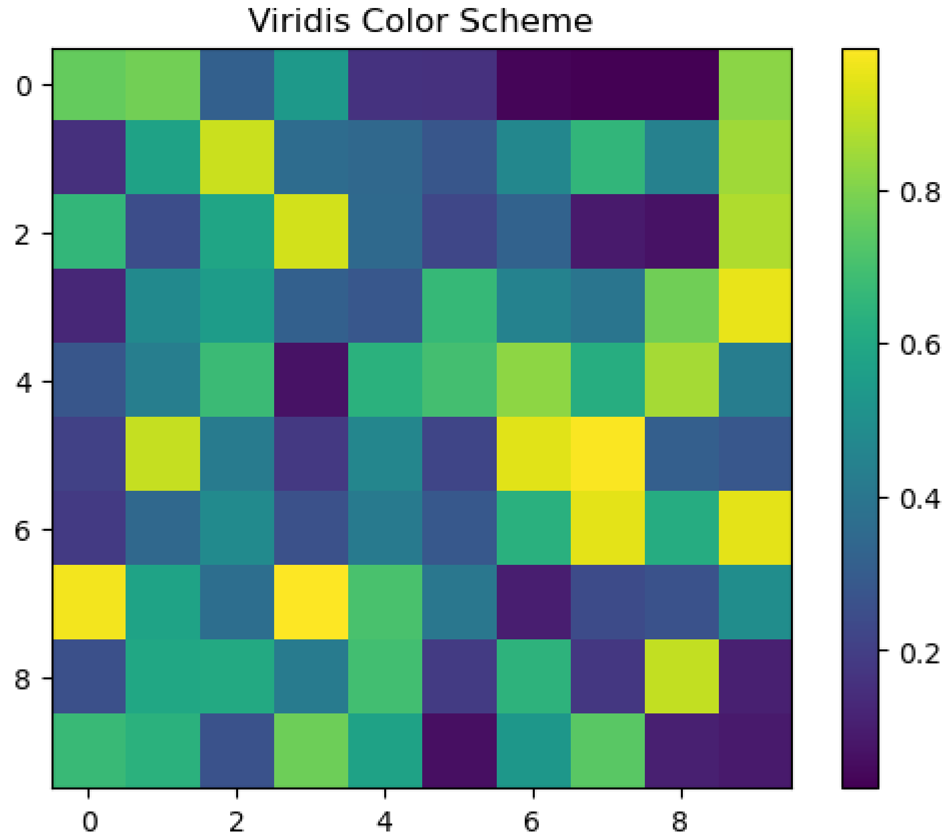
In [8]:

```python
# 8. Log-Scale in Data Visualization
# Using a log-scale compresses data values.
x = np.arange(1, 10)
y = 10**x
plt.plot(x, y)
plt.yscale('log')
plt.title("Log-Scale Data Visualization")
plt.xlabel("X")
plt.ylabel("Y (log-scale)")
plt.show()

```



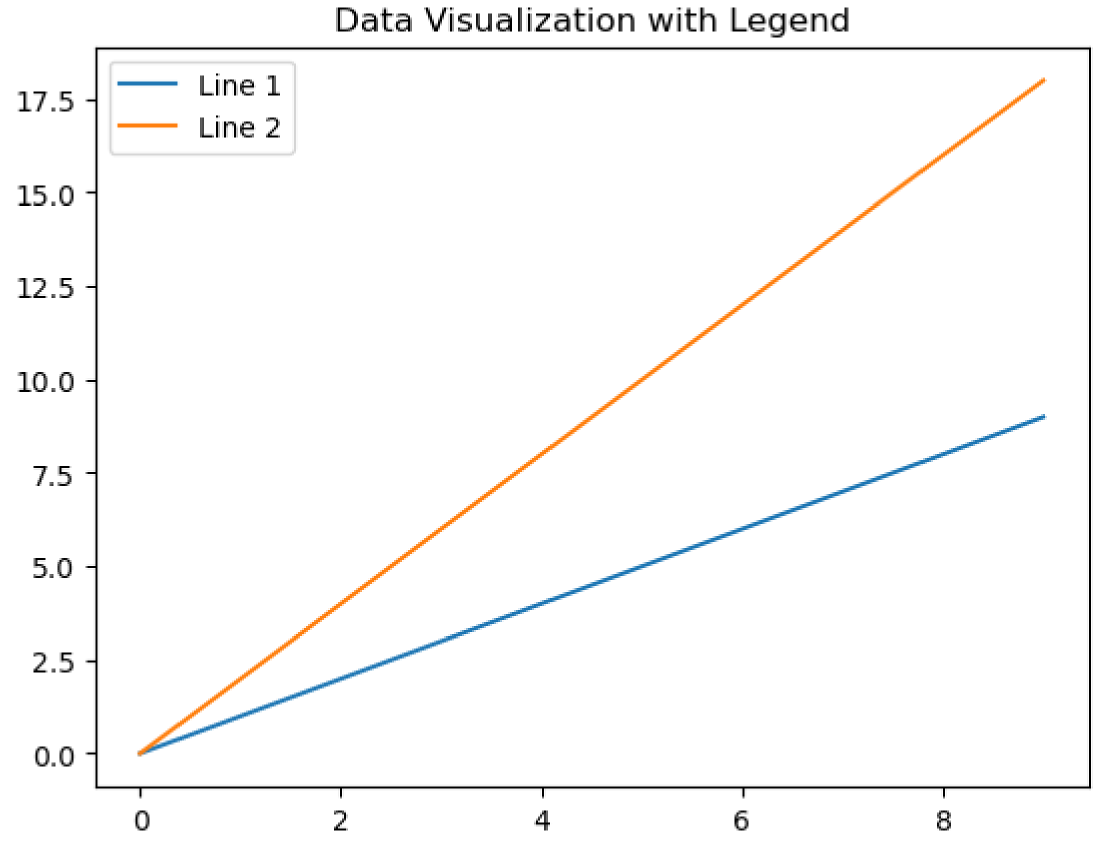
In [9]:

```

# 9. Color Scheme - Viridis
# Viridis is a color scheme for sequential data.
heatmap_data = np.random.rand(10, 10)
plt.imshow(heatmap_data, cmap='viridis')
plt.title("Viridis Color Scheme")
plt.colorbar()
plt.show()


```



In [10]:

```
# 10. Legend in Data Visualization
# A legend is a guide to interpreting colors or symbols in the chart.
x = np.arange(0, 10, 1)
y1 = x
y2 = 2 * x
plt.plot(x, y1, label='Line 1')
plt.plot(x, y2, label='Line 2')
plt.legend()
plt.title("Data Visualization with Legend")
plt.show()

```



In [ ]:

```

```