Exploring a dataset

In [ ]:
```
# Display the number of rows and columns
df.shape

# Display the names of the columns
df.columns

# Select column birthwgt_oz1: ounces
var = df['col_name']

# Print the first 5 elements of ounces
print(var.head())

# Replace the value 8 with NaN
df['col_name'].replace(8, np.nan, inplace = True)

# Print the values and their frequencies
print(df['col_name'].value_counts())
```

Removing duplicate rows

In [ ]:
```
df = df.drop_duplicates()
```

Handling missing values (e.g., filling NaN values with a specific value)

In [ ]:
```
df['column_name'].fillna('default_value', inplace=True)
```

Removing rows or columns with missing values

In [ ]:
```
df.dropna(axis=0, inplace=True)  # To remove rows with NaN values
df.dropna(axis=1, inplace=True)  # To remove columns with NaN values
```

Changing data types

In [ ]:
```
df['column_name'] = df['column_name'].astype('new_data_type')
```

Checking for missing values

In [ ]:
```
missing_values = df.isnull().sum()
```

Checking for duplicates

In [ ]:
```
duplicates = df.duplicated().sum()
```

Checking unique values in a column

In [ ]:
```
unique_values = df['column_name'].unique()
```

Checking summary statistics

In [ ]:
```
summary_stats = df.describe()
```

Checking data types

In [ ]:
```
data_types = df.dtypes
```

PMF (Probability Mass Function):

for small number of unique values

In [3]:

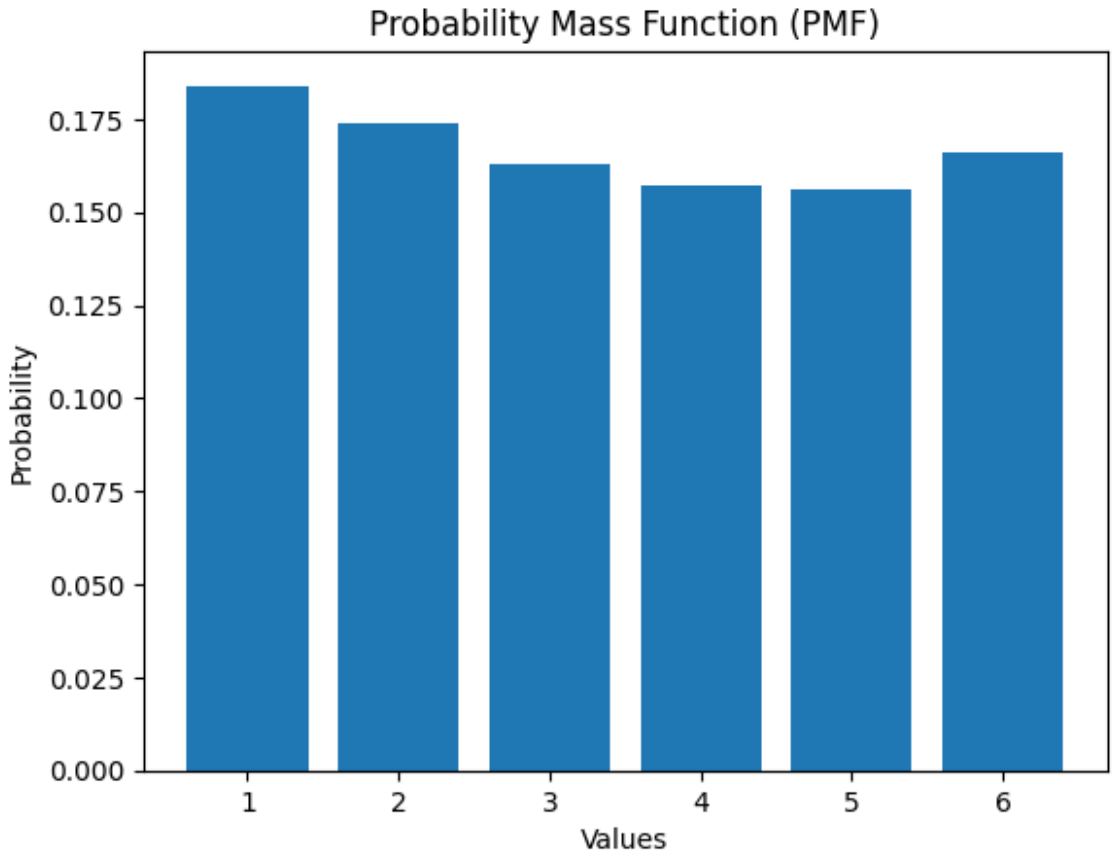
```
import numpy as np
import matplotlib.pyplot as plt

# Sample data (replace with your dataset)
data = np.random.randint(1, 7, size=1000)  # Simulating dice rolls

unique_values, counts = np.unique(data, return_counts=True)

pmf = counts / len(data)

plt.bar(unique_values, pmf)
plt.xlabel('Values')
plt.ylabel('Probability')
plt.title('Probability Mass Function (PMF)')
plt.show()
```



CDF (Cumulative Distribution Function):

for exploration

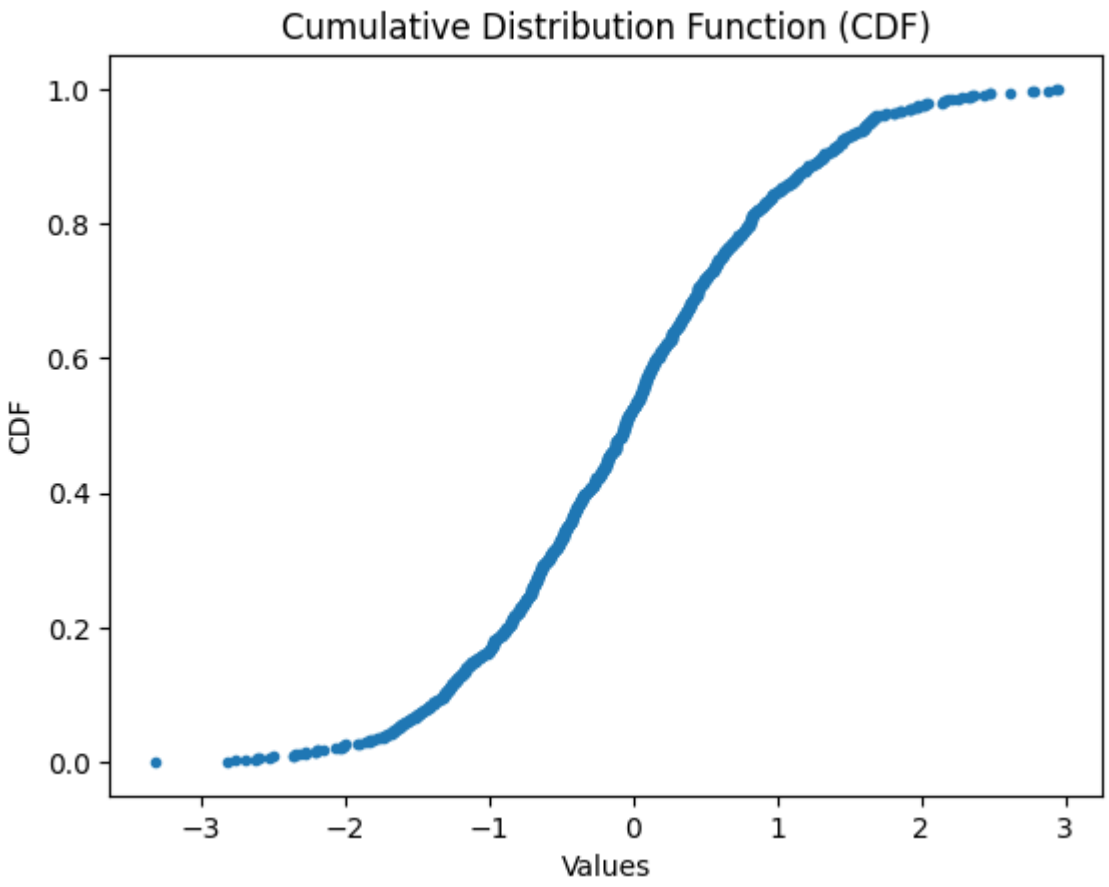
In [1]:

```
import numpy as np
import matplotlib.pyplot as plt

# Sample data (replace with your dataset)
data = np.random.normal(0, 1, 1000)  # Simulating a normal distribution

sorted_data = np.sort(data)
cdf = np.arange(1, len(sorted_data) + 1) / len(sorted_data)

plt.plot(sorted_data, cdf, marker='.', linestyle='none')
plt.xlabel('Values')
plt.ylabel('CDF')
plt.title('Cumulative Distribution Function (CDF)')
plt.show()
```



KDE (Kernel Density Estimation):

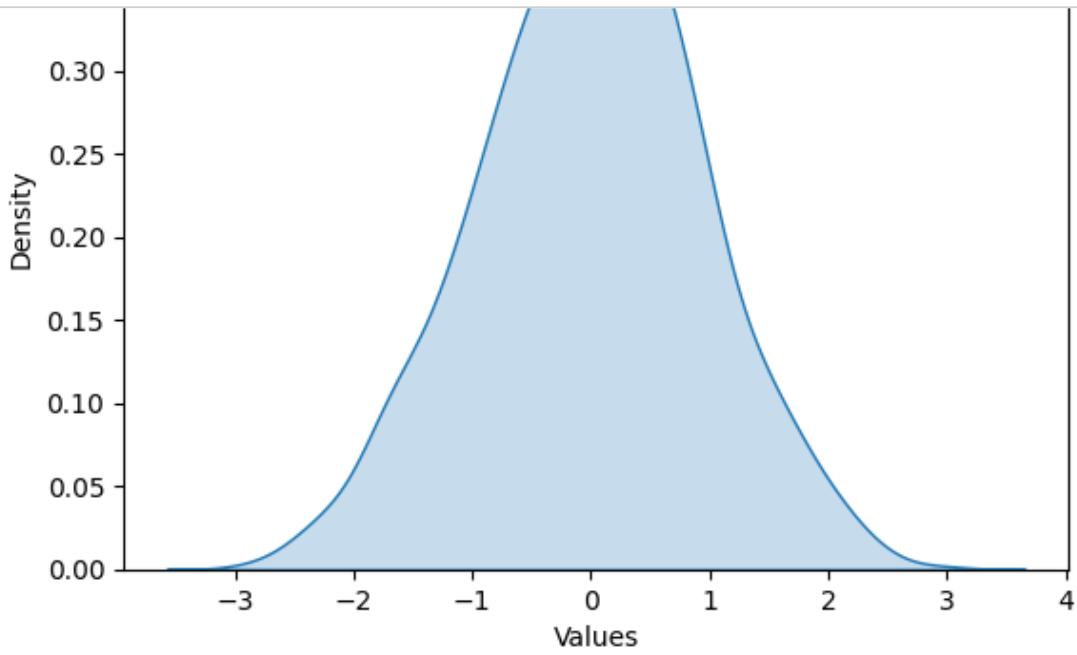
for large number of values

In [2]:

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

# Sample data (replace with your dataset)
data = np.random.normal(0, 1, 1000)  # Simulating a normal distribution

sns.kdeplot(data, shade=True)
plt.xlabel('Values')
plt.ylabel('Density')
plt.title('Kernel Density Estimation (KDE)')
plt.show()
```



Linear Regression

In [4]:

```
import numpy as np
from sklearn.linear_model import LinearRegression

# Sample data
X = np.array([1, 2, 3, 4, 5])  # Independent variable
Y = np.array([2, 4, 5, 4, 5])  # Dependent variable

# Create a linear regression model
model = LinearRegression()

# Fit the model to your data
model.fit(X.reshape(-1, 1), Y)

# Get the slope (b) and intercept (a)
slope = model.coef_[0]
intercept = model.intercept_

# Make predictions
predictions = model.predict(X.reshape(-1, 1))
```

```
---------------------------------------------------------------------
--
ModuleNotFoundError                       Traceback (most recent call las
t)
Cell In [4], line 2
      1 import numpy as np
----> 2 from sklearn.linear_model import LinearRegression
      4 # Sample data
      5 X = np.array([1, 2, 3, 4, 5])  # Independent variable

ModuleNotFoundError: No module named 'sklearn'
```