```python
# Imports
import numpy as np
import matplotlib.pyplot as plt

# Create a simple dataset representing votes with gender info
votes = [
    {"name": "John", "gender": "male", "vote": "Bob"},
    {"name": "Mike", "gender": "male", "vote": "Bob"},
    {"name": "Mikaela", "gender": "female", "vote": "Alice"},
    {"name": "Anna", "gender": "female", "vote": "Alice"},
    {"name": "Daniela", "gender": "female", "vote": "Alice"}
]
votes
```

```
[{'name': 'John', 'gender': 'male', 'vote': 'Bob'},
 {'name': 'Mike', 'gender': 'male', 'vote': 'Bob'},
 {'name': 'Mikaela', 'gender': 'female', 'vote': 'Alice'},
 {'name': 'Anna', 'gender': 'female', 'vote': 'Alice'},
 {'name': 'Daniela', 'gender': 'female', 'vote': 'Alice'}]
```

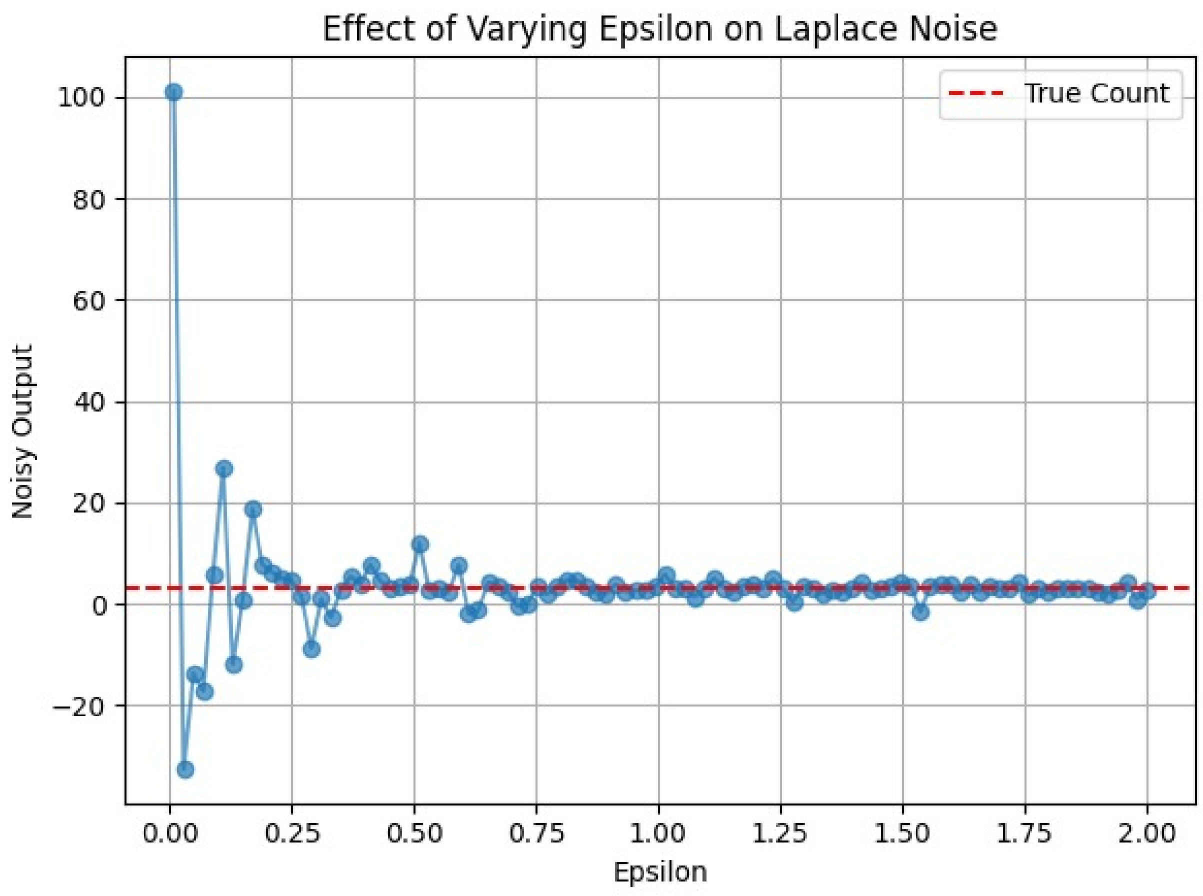
```python
# Calculate true count of female votes for Alice
true_count = sum(1 for p in votes if p["vote"] == "Alice" and
p["gender"] == "female")
true_count
```

```
3
```

```python
# Simulate the effect of varying epsilon on Laplace noise
epsilon = 1.0
sensitivity = 1
scale = sensitivity / epsilon
num_queries = 100

epsilons = np.linspace(0.01, 2.0, num_queries)
noisy_outputs = []

for diff_ep in epsilons:
    new_scale = sensitivity / diff_ep
    noise = np.random.laplace(loc=0.0, scale=new_scale)
    noisy_result = true_count + noise
    noisy_outputs.append(noisy_result)

noisy_outputs
```

```
[101.17428770735073,
 -32.59561742341515,
 -13.64206864362728,
 -17.161474939510445,
 5.707539909495431,
 26.843235246607474,
 -11.860255831974401,
0.7830043826013648,
18.660656772043872,
7.791010455638357,
6.183367930533468,
4.980477116886218,
4.564609014238419,
1.448723975134034,
-8.711696786004588,
1.0185426448362764,
-2.6741713855204106,
2.7919039433569854,
5.400162770832913,
3.908448980501796,
7.829035758167098,
4.522351255387559,
2.9661957121254745,
3.356191535084161,
3.768587223831744,
12.063682912224277,
2.566856015158961,
3.0418239463207626,
2.4472231515420644,
7.65632110672925,
-1.9083844689745266,
-1.2969427921749368,
4.2473532684598,
3.6085703810058174,
2.2655629566413307,
-0.5022180496918223,
0.043224171677999745,
3.353118094661709,
1.9182247238010948,
3.5662925877507807,
4.681974096098411,
4.661361456410347,
3.301536103134624,
2.1749181802089863,
2.094150358801206,
3.8662450015310346,
2.4454396089362653,
2.6407437243227263,
2.6738478008145465,
3.308088809508085,
5.805945801092699,
2.9657865966985915,
2.9688833404353248,
1.368433952582067,
3.080148908576364,
5.0728962817039935,
 3.138073001928595,
 2.3943728857916002,
 3.603955271671915,
 3.9366837629251963,
 3.1008836544220015,
 4.903926619005823,
 2.977798545292444,
 0.32182017618029235,
 3.3514955161594058,
 3.2711621449854795,
 2.103341553100078,
 2.7086681742295418,
 2.241471956368921,
 3.1847599451562645,
 4.0907626800814265,
 2.5441917506482294,
 2.9259165311157744,
 3.6243687554513793,
 4.2530151957118365,
 3.6154341325492116,
 -1.561366172577415,
 3.6071350403496094,
 3.7323051966060876,
 3.8788207663624745,
 2.289295186686428,
 3.9340716056690894,
 2.3984243410545663,
 3.665901126741052,
 2.918510947683738,
 2.972984716592786,
 4.188140437052176,
 1.7702311850727177,
 2.93903699270866,
 2.2569164056626576,
 3.0955839874017848,
 2.9421934086259154,
 3.0461991058906492,
 3.0562990063154256,
 2.256820730936906,
 2.143768651336173,
 2.7564935839968325,
 4.115479527932245,
 0.9135302954285565,
 2.6250559096738644]

# Plot noisy outputs across varying epsilon values
plt.plot(epsilons, noisy_outputs, marker='o', linestyle='-',
alpha=0.7)
plt.axhline(y=true_count, color='red', linestyle='--', label='True
Count')
```

```
plt.xlabel("Epsilon")
plt.ylabel("Noisy Output")
plt.title("Effect of Varying Epsilon on Laplace Noise")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```



```
# Simulate a composition attack: repeated queries at fixed epsilon
noisy_counts = []

for _ in range(num_queries):
    noise = np.random.laplace(loc=0.0, scale=scale)
    noisy_count = true_count + noise
    noisy_counts.append(noisy_count)

noisy_counts
```

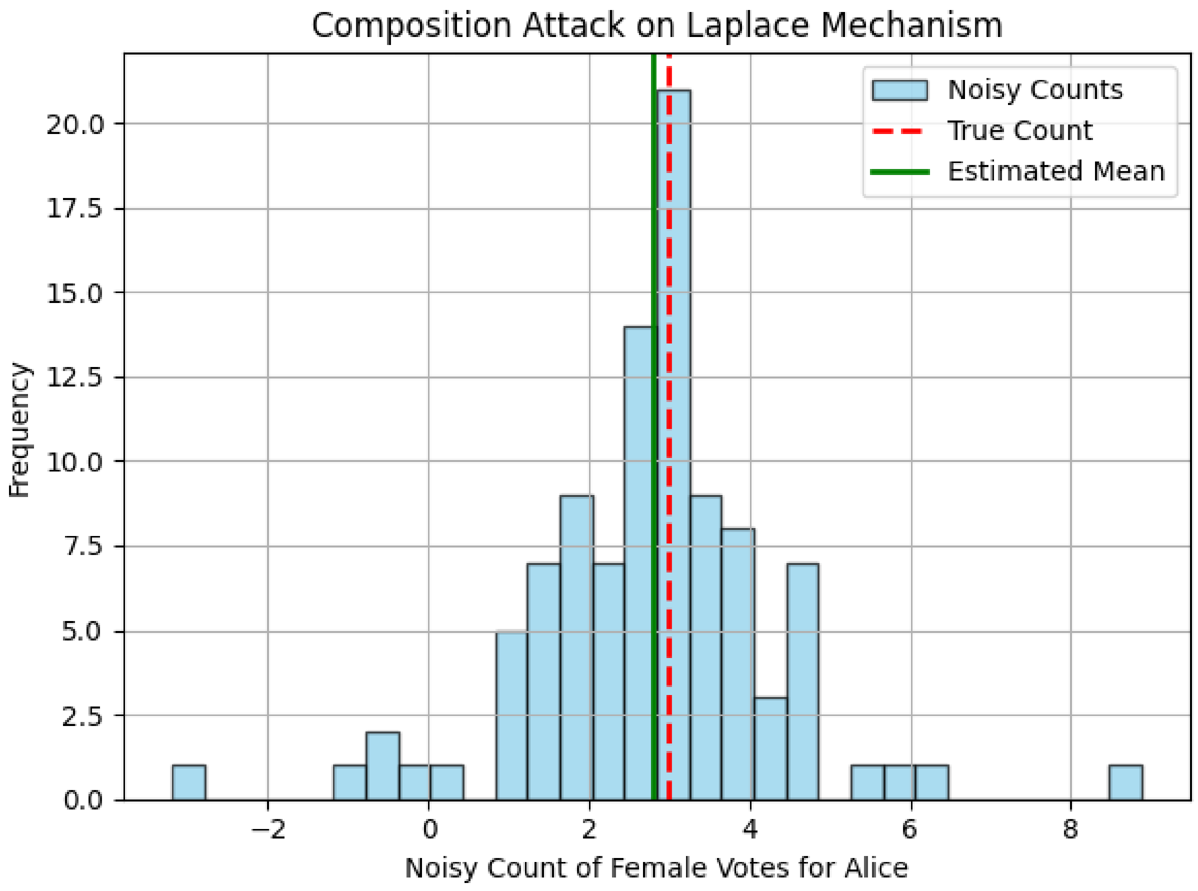
```
[3.2058864368482087,
 2.0326418488396234,
 4.103094599655891,
 1.5005782952261564,
 8.886718273531237,
 4.471795526842006,
 3.573509806733226,
 3.333710953198382,
 3.1439148558098604,
 2.559017038805713,
 2.626532347649742,
 3.524448466824767,
 2.995297367042652,
 1.2249703433670411,
 1.4359065621603906,
 1.280999096973103,
 3.140763986821794,
 4.566996121955926,
 4.713920033312774,
 2.9895297707228945,
 2.8398159133374876,
 2.540803839566939,
 -0.5290026782035775,
 1.0785896363206229,
 4.520415779174348,
 3.7794071743841346,
 -0.4765145000786868,
 3.138065043895063,
 1.2299111399920886,
 4.446490122412003,
 1.4037945261983342,
 3.9017522546595935,
 3.9088847537099647,
 3.5074566429635428,
 2.4443267575451557,
 3.1894589687023123,
 3.0847528825429085,
 2.3812580232848295,
 2.967626897611025,
 4.22059906571948,
 3.423897137497552,
 3.1763299564499636,
 2.497182594918897,
 3.1688076823407516,
 5.975598789403856,
 3.3474426565949114,
 4.017328739826181,
 2.896352280584113,
 4.843781180496211,
 4.740588073049802,
 3.7496274189603547,
 1.8744724748989963,
 2.8266604894347203,
 2.0474280761777677,
 3.822373891100253,
 1.6670487370895652,
 2.7815387714218245,
 3.1290816861895503,
 2.9763834906446003,
 3.09703901669664l,
 4.794418588194692,
 2.455613751726837,
 2.4231007171338454,
 2.302762357161775,
 1.282363482549235,
 4.033024219172076,
 1.2010091912545564,
 2.691432675387246,
 2.947969446958724,
 2.3873701828201694,
 1.2808968672992533,
 0.3953043251554753,
 2.593384412956962,
 3.0516574933754823,
 3.056085721256453,
 1.562640877098649,
 1.8889127130398369,
 2.4288751724399495,
 2.785037799638151,
 1.9945770989233538,
 5.372473218988644,
 1.1662067209284899,
 2.4707522019122194,
 1.9601336984234454,
 2.0119673537103413,
 3.119579537643534,
 3.2639633544471933,
 -0.9861038623936302,
 2.9441780372889075,
 3.9341333545307204,
 2.3088178117820823,
 1.783835295289747,
 2.8937037282850993,
 6.130904896888412,
 1.93104154081262,
 -3.195352435881424,
 3.6481192623106686,
 -0.031396035622722253,
 2.486272134061368,
 3.398626731939283]
```

```python
# Estimate the count by averaging over noisy outputs
estimated_count = np.mean(noisy_counts)
estimated_count
```

```
np.float64(2.7911538075872317)
```

```python
# Print and visualize true vs estimated count
print(f"True count: {true_count}")
print(f"Estimated count after {num_queries} queries:
{estimated_count:.2f}")

plt.hist(noisy_counts, bins=30, alpha=0.7, color='skyblue',
edgecolor='black', label="Noisy Counts")
plt.axvline(true_count, color='red', linestyle='--', linewidth=2,
label="True Count")
plt.axvline(estimated_count, color='green', linestyle='-',
linewidth=2, label="Estimated Mean")
plt.title("Composition Attack on Laplace Mechanism")
plt.xlabel("Noisy Count of Female Votes for Alice")
plt.ylabel("Frequency")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
True count: 3
Estimated count after 100 queries: 2.79
```


Composition Attack on Laplace Mechanism
Noisy Counts
True Count
Estimated Mean
Frequency
Noisy Count of Female Votes for Alice
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
−2
0
2
4
6
8