```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math
import random
```

```
df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/adult.csv')
```

```
df
```

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	capital_gain	capit
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	
...	...	...	...	...	...	...	...	...	...	...	...	
32556	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	
32557	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	
32558	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	
32559	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	
32560	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	

32561 rows × 15 columns

Next steps: Generate code with df View recommended plots New interactive sheet

```
quasi_identifiers = ['age', 'education', 'relationship', 'sex', 'race']
```

```
df['income'] = df['income'].apply(lambda x: random.randint(20000, 70000) if x == "<50k"
                                    else random.randint(40000, 120000))
sensitive_attribute = 'income'
```

```
df
```

vorkclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	capital_gain	capit
State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	
Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	
Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	
Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	
Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	
...	...	...	...	...	...	...	...	...	...	
Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	
Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	
Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	
Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	
Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	

5 columns

Next steps: Generate code with df   View recommended plots   New interactive sheet

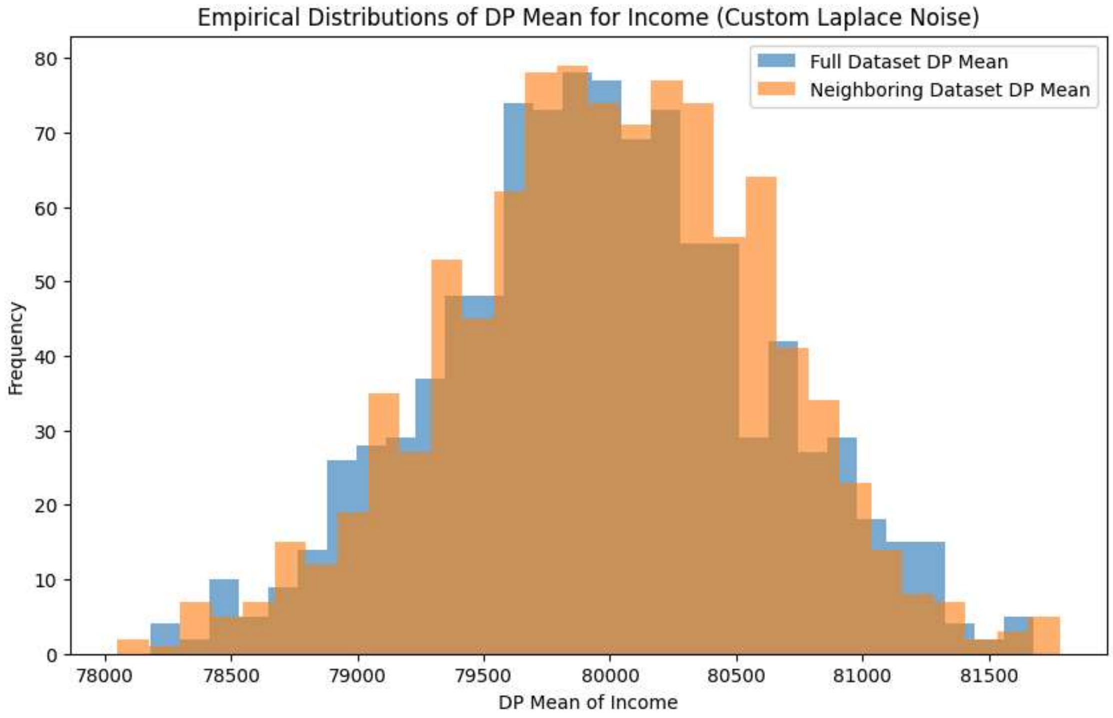
```
sensitivity = df[sensitive_attribute].max() - df[sensitive_attribute].min()
epsilon_used = 1.0


def my_laplace_noise(mu, b):
    u = random.uniform(-0.5, 0.5)
    noise = mu - b * math.copysign(1, u) * math.log(1 - 2 * abs(u))
    return noise


df['noisy_income_custom'] = df[sensitive_attribute].apply(
    lambda x: x + my_laplace_noise(0, sensitivity / epsilon_used)
)


noise = df['noisy_income_custom'] - df[sensitive_attribute]
avg_noise = np.abs(noise).mean()
estimated_epsilon = sensitivity / avg_noise

print(f"Estimated ε (epsilon) for Differential Privacy: {estimated_epsilon:.6f}")
```

```
Estimated ε (epsilon) for Differential Privacy: 0.998109
```

```
nums_trials = 1000
def simulate_dp_income_custom(data,epsilon,sensitivity,num_trials):
  dp_outputs = []
  for _ in range(num_trials):
      # For each trial, add custom Laplace noise to every income value
      noisy_income = data[sensitive_attribute].apply(
          lambda x: x + my_laplace_noise(0, sensitivity / epsilon)
      )
      dp_outputs.append(np.mean(noisy_income))
  return np.array(dp_outputs)


dp_full = simulate_dp_income_custom(df, epsilon_used, sensitivity, num_trials)


df_neighbor = df.drop(df.sample(1, random_state=42).index)
dp_neighbor = simulate_dp_income_custom(df_neighbor, epsilon_used, sensitivity, num_trials)


print("\nFull dataset DP mean:")
print(f"  Mean: {dp_full.mean():.4f}, Std: {dp_full.std():.4f}")
print("Neighboring dataset DP mean:")
print(f"  Mean: {dp_neighbor.mean():.4f}, Std: {dp_neighbor.std():.4f}")
```

```
Full dataset DP mean:
  Mean: 79957.4620, Std: 634.2368
Neighboring dataset DP mean:
  Mean: 79990.5513, Std: 626.9773
```

```
plt.figure(figsize=(10,6))
plt.hist(dp_full, bins=30, alpha=0.6, label="Full Dataset DP Mean")
plt.hist(dp_neighbor, bins=30, alpha=0.6, label="Neighboring Dataset DP Mean")
plt.xlabel("DP Mean of Income")
plt.ylabel("Frequency")
plt.title("Empirical Distributions of DP Mean for Income (Custom Laplace Noise)")
plt.legend()
plt.show()
```


Empirical Distributions of DP Mean for Income (Custom Laplace Noise)
Full Dataset DP Mean
Neighboring Dataset DP Mean
Frequency
DP Mean of Income