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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

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

```
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: np.random.randint(20000,90000) if x == "<50k" else np.random.randint(50000, 150000))
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

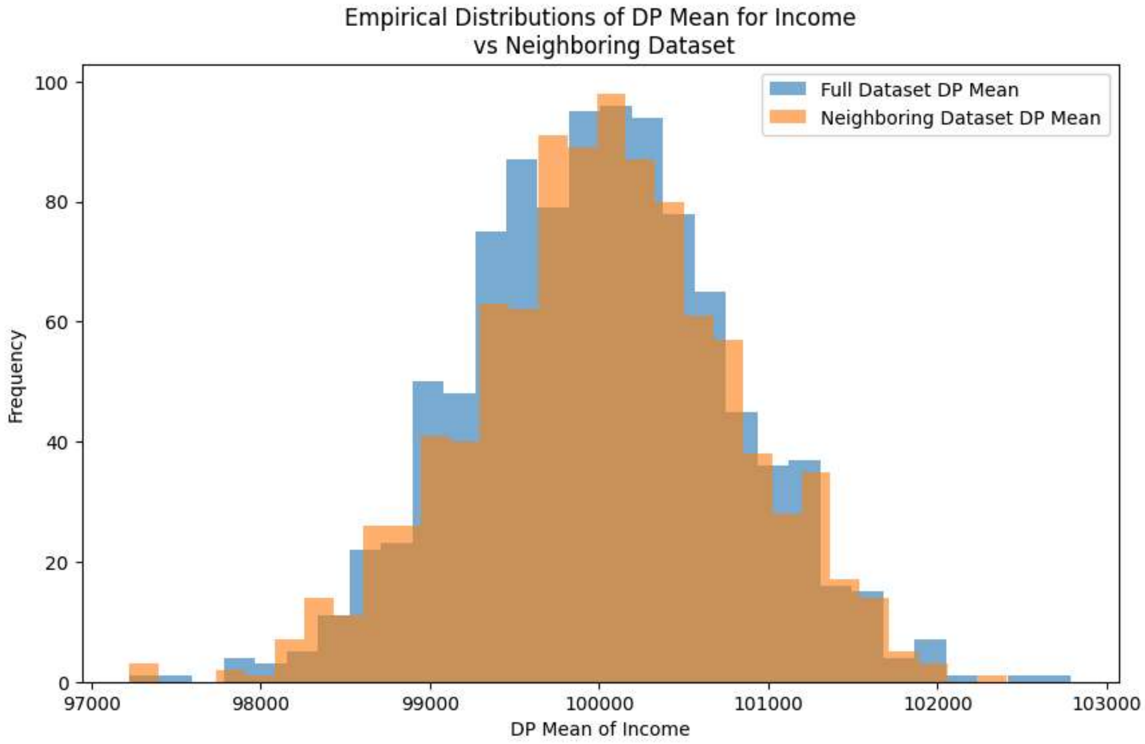
Next steps: Generate code with df View recommended plots New interactive sheet

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


epsilon_used = 1.0
df['noisy_income'] = df[sensitive_attribute] + np.random.laplace(loc = 0,
                                                                 scale=sensitivity/epsilon_used,
                                                                 size=len(df))

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




num_trials = 1000

def simulate_dp_income(data,epsilon,sensitivity,num_trials):
  dp_outputs = []
  for _ in range(num_trials):
    noisy_income = data[sensitive_attribute] + np.random.laplace(loc=0,
                                                                 scale=sensitivity/epsilon,
                                                                 size=len(data))

    dp_outputs.append(np.mean(noisy_income))
  return np.array(dp_outputs)


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




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

```
Estimated ε (epsilon) for Differential Privacy: 0.997994

Full dataset DP mean:
  Mean: 99997.8775, Std: 775.7896
Neighboring dataset DP mean:
  Mean: 100013.4089, Std: 780.0277
```

```
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\n vs Neighboring Dataset")
    plt.legend()
    plt.show()
```


Empirical Distributions of DP Mean for Income
vs Neighboring Dataset
Full Dataset DP Mean
Neighboring Dataset DP Mean
Frequency
DP Mean of Income